Informe RSA

Becerra Sipiran, Cledy Elizabeth	33,$\overline{3}$%
Oviedo Sivincha, Massiel	33,$\overline{3}$%
Villanueva Borda, Harold Alejandro	33,$\overline{3}$%

---

1. Estructura del main()

```
int main(){
rsa_bloques Elle(1024);
```

string message = "The idea of an asymmetric public-private key cryptosystem is attributed to Whitfield Diffie and Martin Hellman, who published this concept in 1976. They also introduced digital signatures and attempted to apply number theory. Their formulation used a shared-secret-key created from exponentiation of some number, modulo a prime number. However, they left open the problem of realizing a one-way function, possibly because the difficulty of factoring was not well-studied at the time.[4] Ron Rivest, Adi Shamir, and Leonard Adleman at the Massachusetts Institute of Technology, made several attempts over the course of a year to create a one-way function that was hard to invert. Rivest and Shamir, as computer scientists, proposed many potential functions, while Adleman, as a mathematician, was responsible for finding their weaknesses. They tried many approaches including knapsack-based and permutation polynomials. For a time, they thought what they wanted to achieve was impossible due to contradictory requirements.[5] In April 1977, they spent Passover at the house of a student and drank a good deal of Manischewitz wine before returning to their homes at around midnight.[6] Rivest, unable to sleep, lay on the couch with a math textbook and started thinking about their one-way function. He spent the rest of the night formalizing his idea, and he had much of the paper ready by daybreak. The algorithm is now known as RSA – the initials of their surnames in same order as their paper.[7] Clifford Cocks, an English mathematician working for the British intelligence agency Government Communications Headquarters (GCHQ), described an equivalent system in an internal document in 1973.[8] However, given the relatively expensive computers needed to implement it at the time, it was considered to be mostly a curiosity and, as far as is publicly known, was never deployed. His discovery, however, was not revealed until 1997 due to its top-secret classification. Kid-RSA (KRSA) is a simplified public-key cipher published in 1997, designed for educational purposes. Some people feel that learning Kid-RSA gives insight into RSA and other public-key ciphers, analogous to simplified DES.[9][10][11][12][13] A patent describing the RSA algorithm was granted to MIT on 20 September 1983 U.S. Patent 4,405,829 Cryptographic communications system and method. From DWPIs abstract of the patent The system includes a communications channel coupled to at least one terminal having an encoding device and to at least one terminal having a decoding device. A message-to-be-transferred is enciphered to ciphertext at the encoding terminal by encoding the message as a number M in a predetermined set. That number is then raised to a first predetermined power (associated with the intended receiver) and finally computed. The remainder or residue, C, is... computed when the exponentiated number is divided by the product of two predetermined prime numbers (associated with the intended receiver). A detailed description of the algorithm was published in August 1977, in Scientific Americans Mathematical Games column.[7] This preceded the patents filing date of December 1977. Consequently, the patent had no legal standing outside the United States. Had Cocks's work been publicly known, a patent in the United States would not have been legal either. When the patent was issued, terms of patent were 17 years. The patent was about to expire, on 21 September 2000, when RSA Security released the algorithm to the public domain, on 6 September 2000.[14] There are a number of attacks against plain RSA as described below.When encrypting with low encryption exponents (e.g., e 3) and small values of the m, (i.e., m <n1e) the result of me is strictly less than the modulus n. In this case, ciphertexts can be decrypted easily by taking the eth root of the ciphertext over the integers. If the same clear text message is sent to e or more recipients in an encrypted way, and the receivers share the same exponent e, but different p, q, and therefore n, then it is easy to decrypt the original clear text message via the Chinese remainder theorem. Johan Håstad noticed that this attack is possible even if the cleartexts are not equal, but the attacker knows a linear relation between them.[22] This attack was later improved by Don Coppersmith (see Coppersmiths attack).[23] Because RSA encryption is a deterministic encryption algorithm (i.e., has no random component) an attacker can successfully launch a chosen plaintext attack against the cryptosystem, by encrypting likely plaintexts under the public key and test if they are equal to the ciphertext. A cryptosystem is called semantically secure if an attacker cannot distinguish two encryptions from each other, even if the attacker knows (or has chosen) the corresponding plaintexts. As described above, RSA without padding is not semantically secure.[24] RSA has the property that the product of two ciphertexts is equal to the encryption of the product of the respective plaintexts. That is m1em2e (m1m2)e (mod n). Because of this multiplicative property a chosen-ciphertext attack is possible. E.g., an attacker who wants to know the decryption of a ciphertext c me (mod n) may ask the holder of the private key d to decrypt an unsuspicious-looking ciphertext c cre (mod n) for some value r chosen by the attacker. Because of the multiplicative property c is the encryption of mr (mod n). Hence, if the attacker is successful with the attack, they will learn mr (mod n) from which they can derive the message m by multiplying mr with the modular inverse of r modulo n.[citation needed] Given the private exponent d one can efficiently factor the modulus n pq. And given factorization of the modulus n pq, one can obtain any private key (d,n) generated against a public key (e,n).[15] To avoid these problems, practical RSA implementations typically embed some form of structured, randomized padding into the value m before encrypting it. This padding ensures that m does not fall into the range of insecure plaintexts, and that a given message, once padded, will encrypt to one of a large number of different possible ciphertexts. Standards such as PKCS1 have been carefully designed to securely pad messages prior to RSA encryption. Because these schemes pad the plaintext m with some number of additional bits, the size of the un-padded message M must be somewhat smaller. RSA padding schemes must be carefully designed so as to prevent sophisticated attacks that may be facilitated by a predictable message structure. Early versions of the PKCS1 standard (up to version 1.5) used a construction that appears to make RSA semantically secure. However, at Crypto 1998, Bleichenbacher showed that this version is vulnerable to a practical adaptive chosen ciphertext attack. Furthermore, at Eurocrypt 2000, Coron et al.[25] showed that for some types of messages, this padding does not provide a high enough level of security. Later versions of the standard include Optimal Asymmetric Encryption Padding (OAEP), which prevents these attacks. As such, OAEP should be used in any new application, and PKCS1 v1.5 padding should be replaced wherever possible. The PKCS1 standard also incorporates processing schemes designed to provide additional security for RSA signatures, e.g. the Probabilistic fSignature Scheme for RSA (RSA-PSS). Secure padding schemes such as RSA-PSS are as essential for the security of message signing as they are for message encryption. Two USA patents on PSS were granted (USPTO 6266771 and USPTO 70360140); however, these patents expired on 24 July 2009 and 25 April 2010, respectively. Use of PSS no longer seems to be encumbered by patents.[original research] Note that using different RSA key-pairs for encryption and signing is potentially more secure.[26]";

```
    cout << "\nmessage: " << message << endl;
    ZZ n;
    ZZ e;
    cout << "\nn: "; cin >> n;
    cout << "\ne: "; cin >> e;

    cout << "\nmensaje cifrado" << endl;
    cout << Elle.encrypt(Elle.dividirBloques(message),n,e) << endl;
```

```
    string firma = "Cledy Becerra, 71666666, 11-333-1111";
    cout << "\nfirma: \n" << endl;
    cout << Elle.sign_encrypt(firma,n,e);



    /*string message =
"036713981774726449951121691785546039832472597384630845817817419399289336436327086
4736290558916979477798552154396769866623502179007760951112779738836588575095459619
2821113235791506020400616627876778280254085606176239120946645064696033100935099149
5144835175542022069895386873293022634815340900629301348714301257731461859349719499
3654893352727406288297155561573515601623974559793571861356471740014047338407754007
5708049205474103831296731533917497491057615374181637834731492411658421894174028025
2696312318736031743894401061414956545836734000336448020887377644592145187465183393
9440332712108679222062626072176424760529133108516881849467246786031275104893943485
9257068505494250151851155830311428630292767711540813249484358669537029423282256674
4893048373944141115129440913749070224242963728699331556750872580909238099029320227
6965930046245357769466843944515010601385567102560095536197388767785407416483561707
8018594943444535606244765848508245861498953855022568748238227598271438323623211473
1057042672933618059046868593382318779079180670621287926717114880362425022592053354
0182072857818059602894897692861048626229034099830171241425855229954861037311665716
8571596323769415704798868477718244258878532107806143407852069851779580100746993937
2795313595150681104078666579645685119393598674857145804061855350003030431810327035
3423318056960322005743945930223918294232998979968111525523564726849398748426384158
9431066718944271703689368016913407299413793910140392910348928151860627856901274191
8737926881470841611673415107441704242790414805336416500174309719536056215737526567
5423235281100323978249173293527005318498753166480664389468166646942925621686615349
2897934006350013576726233508136021031123474904156483209131903692169627350252291441
8971854638785652192215787500989181971558729772793736447265863074858286268551494983
9342698422360902705509368090356610429644123726721276371303929668728415328521249271
8622063125719480158303492987268357593668666828746410722777140811371577985453678679
9906819330261270031648153029459933343283198584254345869194537428641617681166085400
6139996798645574939427322690634529698608421618886391456341933822128908095913006201
2123240642491903912866914402917732051226176484278924883242542056058928346174053917
2474015074258402699722883319750332153542904995246532582884827439407426481302774134
0218268085701362704371162447675204137195780365167330162209677032693833664796741350
7260717524034403036119973488361275862868306757058049358646967429259019578019275112
1790027021265171281955972152546061260521370157078367881885054219178543075220010170
7211465979558444490305141535701362329926864321231219116070873435539840138615723648
2759032663485773366796471733793141790557236219008923357907392171232025858566610510
6589796526239776514185860458782109304464291776824055158987208705278646763423265581
1067426003501771778796023118471231889666385374762061010002160540630601353366034299
4378153532520686555499030697807826009041308972636562559799742386937767830683690925
9456973000700650978993799934839480207083552124287092884967270085016915240690049907
1942551280730502915056813052072320758229509205650746050313205627888634854855093209
6564823010404649969298828901154401917043770756686516465932183266683618674639135168
6593461200314907935713570665437389839950674211019697493926883062804465714306394923
5012031074297224583259051081338947894032168135639913647613611539009935236396251100
9020724197842549873954568173750637499329831687678975147119609772728119739890798483
5080974436862393890338246048736777635936415213966187069120389588289472930069315206
1524748076919828009179993910005575581394968917357664373008053441608864849334696306
6961810874860566414160181340702325705481979361252795076210623675378680777302450803
5292828579601133915769952233088121778076775253604581716263869391508880166685981780
5893408609054886112852460572037724142933820961596967472860432661081208307537478143
1149577905692261130683312480684587790752007958806137765373045831357796765522586870
```

1576832703473849379866628090721334375443943121743777623028660270243616512955949874
7580837127420892570468174885777238734684964548943774617073463490687582696447669937
7153819254803227229975233349431972299541891890122559704553817116341326225498339429
1783993679581536524517567751134926593974233813452500387309767773718798634228070275
7612634497924450298848028060860403987735465934694375374911222620810022932506162794
6660026569976679170949365179496165179100659304442029727652875100462463778500360142
1866429921546836076094996130927205054260841735364386991767351327422007640476017513
0612687574059525128617589223394917991086470506076319690697140299759122135284865712
9032775941855655847194159772457176379103595386342044465237192914757901561851791330
62957717415301987194624442235085956377063489015097955791478459874308046589913687350
0831163523702721669516232237930193268609201397175443523011219360164987683487468703
6388849859917474388253682980826364068162408461703433177068280950454265519077056632
2556203835190847913971768830013248130080972402975161935452279201337130778846822638
0765277998339627919210941386754868866669200979753022099160552629714882237457566855
2752383810320580150962734442701787375760188192324600059475306729925833264781251172
1133514818257579658920135825840076378708685249002461073226239191478015288915915660
1735934825101177746240406784652385114000309961689070062540815958726314966616393068
3406893502374006491605187397237299453220153468198448800104879702132910932981948573
4033579460730387653201651738056519912347753653541343246071712689651664326662808295
1180689337660956775413719468638531039704773619774035308372721207787378752563991950
8813971077817958010476028698486111268693402871162397447358605058266516426059666060
4512598298238763753492497821405615566433564044042821553157467818919476151255850499
4457339059804132516219353968661624544009589762583342679989680995931355218882108509
7869443838666717381359403142542043879265323542346372665503837054172128719062398898
5504136030622038861108529025753227196081040474640842028779232032416277399585292003
3157251748033249003509095672566255825518416952588038878918164324781376695788522607
4037775550598773799660474677783446156960509787415992965196449132941711505143560157
3784016181223774710390364716317739481088626755559626236464669421600624626825322667
4640938879777924196460015829648196818214817675221194107255149608103335181725809934
7755819270764342522236610212095423608379327772108992636034990538329608103361047963
4400142740254605243384396742955783587373594355768537485108410499871795913973577948
4696752063165275826966543618471218505656407386165488853502177342592084677316721888
3924881218045478748099730200246616183301046652524485111942942390373786963670926762
6641225334037567282166999162812247479693125907077349302692524324372733788697786934
9033859396107458701419590363486750798420693173704237143338421098020290198085296969
8511326022120019703092999361031603970604222845120110883943998682981592581006068154
8545521934119395640445836644123668706135380666701009373264705076038559473132766019
5241738905079199235544782605968269431346939862111729418869614555243601020748117007
1067714299184590527384084012055059684709832021417870223427815518642215732933430539
8377902143330262597075186684084169041933634840474434952040380126818143063427683572
9797493088064086810032567386280926753811165537868346671626614886944770323528661237
0661742725606974983586827874089959431755729617524625733210099528938324135795990562
7988772402694351091701919073709929244161019696697498737912221722168092465967907965
2406854948621889729017641660421230076991686103780280296750677394875546988539997911
3758374462664960670044705246660370934065357874207887836562977812184973190784885556
5977706309158651151729430268370340368652002930999450086503456268476801624094739526
7655776600389098964628918294645067742441035408738252383786290404862124186982391672
2238846929849698586189434321770681922429079298866731800165995118303156002140066796
6001523090837155096678839789933749781426323699135175094283963290173772921091675257
1574135788743864950305396642609845173734772607071155581254222509125004677756290943
8870512511978352354194376413155178803973063855274192467250588195229005134885227847
0686182739069537499422740380269988560812870119629257104874554716832726435878799001
5738658655649908879621048515818038488404084506527655276942558755550628765813955438
3289601031078108992447890018488397087367630880333342201667201584929147919309110060
2307671878756377874572666749495373484192353209603078578547346233089106178236621593

7355937327233222261654328453764086215363843536159286198336918179701961283061593341
14784891332902131951260338646577367108579442877470669959073809997420557001790515 22
8460536107384521223821868186100946292177028353244987823490253097870118961087067987
0494822613502225795080025416800092902277219409658491649800563992596094946179271090
1984827646104788356990409679205168156415559897182851115355431114770468669056418766
4213502368247958545277818553550343533001837348824534120244149675991155724806837256
3094673311609193175844543647540628200984795723379600253157298047877553363275291373
0602034973840246957471872609216985092902641963169430300133706433566897428664710129
1834847206588382058734588630210000539605758906797004888018583485674192601342189142
5112016974128358360422253077017732036629295094202119910955575500829658681039668453
4017127030813629607855621900985901122519623435233520410729035338230028755231804601
3674789100777246806323110259253192315377237500076320127032438232337095179228121401
6117165380982312098123461760970829263070047090975664534720395460306534453338087966
5447950382622074767829901362501699367810981324253968182272752328332404009300972703
9869144671090285843699399188012415048777803240146038012964871986103945913582312198
4026904469962236280306134922001479474892108393826875624455671332950818707270063963
5298491543168124935237738129631215867613253641087211808207050699806398772873346021
2179985588834264669246828117320081418023431445761057310494288537027997610780429890
2464089694696856289664985343357032851229747398050476591261543269146349699588504790
6744246475869950916335695828238024486397633602182244029344578493671900669363542228
7386669262457420491243946377079625995420363654449647850966518743040427102497768192
9507238884162720192711848665326851365721325844114786870427031087240244041436366824
3866585715051207593442888886343565051654768804639866443131777038354543807541194511
9121443886768642864645112381857457008277993202586349462048891390390399958431600090
0319240851647184636625218005563886404378108187643263815041164757470203102003048193
2524647421741660904855752715858933070628665617456012182421109030174820789894851007
0086874921431193774299504038158699164021628920272648325661735174514472919063561166
9308837715732809788909459416723646766953857661792940999043067649563113512052912394
4326428194402801005077931621887523424145775269757447702913215415409661793784760358
8587398932402778879352873238555332360630166157265053895882069408861313091882432964
7386126858050483243663301560703616836764409936251163508410602881192159798403557885
6416711968744800554887868183011786734415780584778795123319053804246555810065386962
3860394506672989834837383926081515914095692789767631363722424628605612013053526016
9808621274400983804211726127831163392790512608370821333730447399821261501302593619
5184400572550584545997167289049773500008152832915591293458584764904814744099867711
5103795026375299674175258425882559678263688995349181655747226538090890984930415078
1390153006747627031982352066183377332036249899231380284409251316504309692080660276
3246614787046438091623397683758086913383410674160071833737958628000756320841545912
6302591998586457901423297018273688182240568288017988474558821962864122462598407309
9254454080930076133965140581727032995757201817029480129587444570319618786978093709
1977165867370403343190981351248743648423842423616345370150875849111839374335967063
5607310486172810836249443652509146066221973536393884898199813747306378590727989119
8801801938511131032337829451304506324933235066142287391135942703807173916724447697
1237687020360781580632805782593892879601518904002800696839648677363077318020534702
3949210010314533204455679229232121022278353408429848103569708656331788591149703978
0783950038239713691713117914567951499952521487882347612497562142412299425375118619
0190773872581335014349381329255539603536739548898127844475665856091016350711877027
0294744926367425306799596407135642125919702344576628018128982756368941369294484497
5406427112644073620895683500732073766362623796125863686536044944506247018224242262
4924685685953539646350541749235020914512149234535004247163039394507189414931398297
2593038670983572181281376241783842831359806344365903109254697433652217457379026143
9996508036156169728343430727459550749978427134037729324398338291515621552297929118
3510376923956935088760514840458544810238384728332301259779526625561159425741653780
8612761345844455374793842531189892403964438001541203053642468391944893050900364489
4606970041114450333423759500615768175469365738814881447486147592854663816668642091

5610121867077720195973507364763347340587127601996654743505855927547485647402657808
4460266360416917453614475899642047944977274939972240267900275533839619576576182732
9138583582805346544008331620488688755323429431204866088183449322805713568033334800
8547791068181027340422656579673660645374125486058970734146561306449055782867629561
4822312021847974001339999227102917157416125525831722924938938341123642807496189116
8373711753165856064777210541191810417885221534000531840545116099367691381914529734
1505146004550420692410681223846516839368861587537283034198361932906391527017119753
1903534406240664659453158664303818236096939950433917645342972788688297498409514513
2498022672651089120760211420626447231892182150172504993066834492688435923964530012
0315929813195426432069164915853484195534591641939505707446049576546513543648831821
2304341870826078280931914488598496418717345514831128519723609282927855061031021487
8610035120679403234546495982477215865912630354683125044131581228442753749818755669
8739321024239977608995083407827202214206816483318478629832360369481807199616641302
4685818867830487968119201665620792138325792053952852354408865626131299737575903989
7001723151105798990720810705801705821875867236112089216613515457632735025831373883
6931990004552954209788779038768047072808475766797232606464329000538791336041176549
4428615355879296450549984523756754678265855391460049689392528105275221830115735740
0991131066467896634083814476480864701418374841412667429464362944198533400172901507
84009168918572552303446787090919260759117218432744135180529886119509998985205273751
0558976717833323222627732904086234625955801731209382375910319680695431508139850499
2
4246460323092560198912341992753028551728116128367526269373357071356813742620509939
8714257445915714027981200925393175459574612321037820503287444006701257618353079602
0552055278031006915517970421247149727066129694142005215294933235053094927742813010
0215133682696180643075360038529895930747048092231259097337873769558370497770666791
2525550076252827362156512113994084055757132176745415826652307366134225292527910404
4168672354035615104269586559639002010890173539749569280611946491915369458586584141
7562863329856023511492530049145827643623757659912920549293888956944662311516304741
8858617211763094950450161132205405266485625966111130786192926524445340686421495051
9893089684303951878570595010010939852227572442032775045887770960874002175598854774
8796913371584128024143394506547160789395706695591959987431098586546013875536143969
2085309203140918377047580909456299302186953548234560836106550000304918754260514848
1930406510713616832986821424492625393373100133258025919745340758891653689021928285
4830194203980137503515144699098479112357014333584283669625863798709227844907750900
7967667640441477584843166244213888667290130821494376394577304177009007692811106966
8314656752930256326207327970522848719640227454579356733022996016835479137981565999
8717981179516505958151242842621749775365896867713500755842717411733679110152181635
1691770908952957238039382690256775544399783849442674815815133864766422766205982864
2857221965025449337543489737180731297125231817099635201546176666882766053034143345
7851395651666591590764307171860683986404888246340911540932948573379264168049348062
4916071821071414401884191731268531902653600179309176343157368884628450208808012657
1866350006193414530027512128444815695171903425075698359978355862231017875054625648 4
0625066876638101294279751618416686949833775795650496246576777204451860424798297872
6931679741786589616670843592690177901499792096353252465942333728409880176207617608
0133587987368663563343692465449015793640140244494043439183772711522497842580439432
1292172857217836517832127934251030617164644914060391400672314826097453808704051471
4427175896308706952478485064962290392701283623405611910073454725439959504276861637
0949068974535773536263972673594086628497389969938709637435416821280520531886012980
0793185616004467567760675114594970734448079420219575864016441405893641995029206458
3428935911375028113801075796620170157389481004248871528456706344210549626795233824
5534762497628896809786748848699841598143280821392875090810289965442317412394229761
5369506239424409298067065248345759216344296812907415791220594124535685359020428184
2559237120568878825418250604232388805195076994515060744288540593927820877277856613
6686278414028735812707332999562397313075416118118228854851320128028022392512845057
6526804926419557524119149125034514035639354512707583464186251842099881847752624434
7174502730131079686950503020442038719588015607051238304166634042639990572930261176

7282495816480424329996432322010812304567509330913517081567727706056018198086974266
2026482770740091461755410490908597777146029127858858360525342834892520356798127775
2299127057612534702359439306985777080407456001585461216783560306724239963497255070
5056319980263186381106369315145165814743863990420523105191679936371083560005211153
6434631359105938408530428155485618930461472532548449055942593808088686824269681491
2415368275350782000403799787225586749268052125236578488411262073550447601214198058
7531212307048230090884803255286257534764131078277185646283324805913663212545601866
2347223356720127513398396001506573456361635624599004077184028378580291158128207496
5956067513934123550230668486616922099549699472256060739865139836462211857502148188
1615601919024742649201273321853294987890724381487141664360600853973982674337953442
1882859917161308420347577933784927742920328881330583797490210138387207682716348835
2115234222248790414504914009127490683076964038507325091028541538303110702683030388
5999631896020489479553238689257544997545987057702938013461558133272465325252674770
9419957218166274469440091123136937529020308128735613787881253082726340615641547611
8810932567234000260169334400370548308017487401280967512939647548289767968260155539
4516865544827780824334081177310767127478909795150763921022381954065246429571796479
8471583658008554043395642749224775770564499149711606937833469398221750092557305249
8355525262237855694908187932941360521410570566574554933733558971520936043128440609
6607167747697472098720984419849485584476180849769316281061805384625393428900506766
4245446622451835893125370461520705885941711080031075630558409966869781307469054110
546028273093530815589l";
   //string message =
"0367139817747264499511216917855460398324725973846308458178174193992893364363270864736290558916979477798552154396769866623502179007760951112779738836588575095459619
2821113235791506020400616627876778280254085606176239120946645064696033100935099149
5144835175542022069895386873293022634815340900629301348714301257731461859349719499
3654893352727406288297155561573515601623974559793571861356471740014047338407754007
5708049205474103831296731533917497491057615374181637834731492411658421894174028025
2696312318736031743894401061414956545836734000336448020887377644592145187465183393
9440332712108679222062626072176424760529133114346755846759436295242888841667897708
9030041177975483942434715559441073076975991391933685807330772025458408860202452668
7665479669314631523075033078200977052013203945629511372483892894884326369561458372
2896093174642380409166350047416330497884937027964289726934501777774369028527354666
0417650730626380598795679759862885881250345251242104258247010682230980738233300593
0571249507297635329556668483022915170715008915598917979337473205723039412919028754
4602464032953415215826478051535299477723745488584926205892453706182053323172180243
5106543081703365818887069400215278241059307906268186666782025169804119925466857197
5353113671103046708871964555083721850688798220193920274821157380509581087660485662
4814384882350475532687071879825947166534486088036645021692822109965274417796227933
7365081271920553606501322229071530088254305127189893030860810576247466552832421550
6844786611339111401688452303991597805096440751315729244395641620905821029848509379
6262848388881148254640293072476402932547478903007944586194026374201914857523791389
6870631138100243962459866258191018022056745984621940206045501418849553263385980375
0901131144075466312659818760761527282844701340418539647859333422566967900169646960
7657506600674830470973105997984149459229700691190778048214141979971121486655207366
7700955435636349012336496159731262599622450005660423123056614066881217814389551424
7609471581890327161516626999774045358428324887299402564208856994011481014339623585
1712969885571988641729357701172670597094590832243296349929329743791425408892661499
6953878963454573245874959470616704385680098085087304365256957880749323434015555382
9708687961467129764717828738874889821406868555211889766485621377156777347082872503
8314026834389743021524092946782390616242735298614985207311810091128037526740668388
4443112788090390134650631549257789374414181257976675911667932240590980293180617984
0081283860130194225947685573176242428650638232168312735831661858428168855866618871
6871230341079625389650297635607434888105398061342803397727665357199466661531925440
4681137800768720080088413211166334220571003634022183289804779280209526044456141321

6974934025900478145647294483346320000812978385214409374283352535600857024822762467
5708371871211797598031502293040147351788104999675396755693576942257268514873437961
7037414958665672137194543336833228328558176791493024070953603062275199019083703331
3653752350131325827912954135002306257240973987285749426727794860957106714162430957
4630859010097324730022906114768460049554027838043681016248958868123904525573753504
3602861049724599466451526488544225134786461863955919301230461302676840395003708593
1497151552175226700329870693185401541138437439643946357665828116511495443453312421
7942563709084592579328850160078502352376104604637661982027229973550396206669592617
0712484241560087001698961357100256589103983944051936185497896103178706667029463275
4590152782942689263234842673274962975794679411768892417107347513681865667544118319
8740004579587285557515597926468394620344715159722214912044243869664841305891369219
5271513201104169528834823674291249327941068363656493882826428039590967780365196163
2770136547949028362712775182364579511893505526875666048714416628721671821719604095
0072461610033563127406642761099377711936816477192144417461992374942202415665184956
9644826454198742498557936836799205612603458676432407496272151333384221164523433631
9317263550027771170628035458136638250252541189277013380678731798970401636338693711
7237116430683826796936392745237999973804527210543856892217886524503584041030385563
9468246987871518824870477195124962878908207790843818341808464471771312740719649603
8792050477660958504957512199452334501965562840843607578003640163767112367167699364
8463021819484952355047517065520054667209703957477654794912359066196710980782106957
2127688723289631802350940463320027167032264054945192364229496424477027666353539884
6212950083501264566237418218886514627815501054588973755312424194327276922580288778
8464844860713913582834863240440166579247577893651127252060249203348063214751471856
2806953443908159823656611417896711310772106040673089092803336224429125662224433806
0753832023920021532268993229273156919701028330302349849620230128464071062325520707
3965794764301434867052672858306930338820607108594573202204946386918372149948793870
5137984510327807043593465125934651639881584603551471848516019008384021889145124267
0690789361506172103746536695068405448882385854427988097423480313426773211426192989
7027334563181971676115786602931247849931205071365111990836160566158550235108718228
0125073843970225220459026049159907931347525641011524044926889441678926224099704164
5086177634819610140526501398540667723281049924987941228597172571526030428486421678
1163632556563638936564339792460732069305042525824313684780124281233447922159197677
4133127508159834465995778720047773097426449306594339210671334675533301568600082483
0173973376900896895459238456537810070243362284866713470572330224094271726267934144
7792467265958431753849848874030614899889426160419206841135860611504441805563168820
5276860263130373524244427577315231958162656628585548155031851199250023708737400047
8308617409227426563915649349319311739283627806642744161215689222994103950434815466
5031437952784307316214231852895967007384166692292662880202756909777768621419466712
8729858324957688222619950090153902131610515457143139524021446515593011949626998541
0866906200185306159660374441697244896031556037949723351493626297668184013842760507
3839967052877340927239586216034625540698567767420398611801275655330559563406678530
6095877000623926432048136811303413181831746415912724944942699085765124692759291136
9518158572155843860200569439792982825859118081237890852878692801710444521101903661
1632004493288366164556775320538491599747813333271970998986648599426245690899734247
0334522834495600052966403149040352119160450357029123891727479363328779923661581304
7045205940709501357946630649547970660054896350408492064849200472247035795537024146
3453323703159439058048003296785717698486883738160717054708803288107830592854391798
3102634392349851072210333721036259905595636793383284308162820802633607604770736652
0732106453915950316358482198178216304631198533337736882558041088020003135489513701
9280025734526174300480117120458371410914932076098151346734577283026206188547191781
1334364628278600839620000193523304434551445921744188685381371897158473010802946674
9584892521371078587228989890278060265185628007683296105612182994213923408820055586
6134553262435632894871152500308674467344574002924243273584691586400253537744286391
9103661916050245315569270820359305621800371970607180459951538477484373990237173555
2536481958861319667166449446515340290333324862048342487367635862483489602324520714

2344290128285844361045699019308607246869712183814132637456055020429923628393798855
2422873154014329816710211428992662667454913856086960584557699273575185888829826417
6554187931765220437697380085494935187564058077137148557605891133218128837840776734
1213579713994146510281956524781415314706143574304946373977416803304313838766846428
5136122133675273724942029540672851500508575010090232346762328388014215865126668305
9430590147332254221901311306073928416813365599557327818043081146436767457266572987
2028496458204747535959750035566467783510091310837035041843179414324443138569365228
2597647205341996286329412080203047792437866987384130899383449226523382965166706939
2984673195975107554391780003307276251622916299302262999351376566732256902631606973
1773142751872905621219419141403387262203328044182095636135223272930011054294077137
0866065944259478000364803812479897469140490696272210135149510553905051508687115996
1869875444046979398260996021948405732814879201067903420829255721988414098076826734
3106337705688332840579243771901299609980014216397907631196510843711002404483320507
5353860935462405594808619300022085072720478523226283135876455913688082878879316661
0166331297429272391551166897815715188466907614406601647262320467706023089041149029
3759516928151760516647660015068762912896095730363137076083446011246059245107009998
6528119778081738331099399311520513740338686177283981040095502748618587149483045288
5655308219685536948553805453307804146701876026353236208080890562618429028229283748
6611372038271979954869977087862519723147202151123249886123994363971570671492638018
4406660485671068344055386414878271640623601103014516677387152280933798446887823412
6362806588726391806043267880168756777997003483379549618555071252851072452698617230
2506147452960177448048902183409722385667631992975742391397212240341231497910607940
8821858944036741323438492952494196659335268002101472614161860178512469470279769938
4023085247091860653554327759728973407455187626351576290773880256084736441115656642
2821669481432207712450209743536561936895967141724759629283561341860620531161992262
3512215199707427431475689452667116965376615080835563720061899597305411449707323334
4192026082104512055656583574077806212423040540548928138450043146163429811590129716
3205813076227382924854621475860230852887215139253059601207210813868934167992652838
5279808804102408407431940388392645881844134311719649760783867785466756170572701596
3027139474066360635712149599714489917455523046147308639165285485188194366331294968
7011094355616770822135537881777637746729044590552489687140778461701212403915562283
1051038693602810931528609794369939703711977944363922520382288418082750964355601765
0397428470158721607552260231725691166065848018570371019829859730611285273419326374
4545420732789948517282848674842862453463095095644201228200091526062389970618627854
7256755279411578846872902252773091645498474986938541091876884323144093996377426634
7876326743923232128237922387549939335368358049110282640403494453141236694618101486
1049067506825213813565292642256014016615803177242800490564151632941397111220590948
2274046268948056888641698400602916954808257948454059161859521226825085022441511475
1240704739041455450206407484276554715276540755592828526765675558832026711383309372
3223238512566801059378641136267245618599143785455394951503370968062669616083053708
8650906508920709598286966883192341803289098126995203422569297572961164378653803163
2624561719174965619246220553472961223000308436655774533574129979661935207011073115
7941195262976986694534306635365865783346569416258154765594686495395049993793357551
2197357883652271332697577886068005725898385129304213510526750956585937243302702697
7670921904061826192842187639677813752291290836713123809867956928242206140059327625
8874209451339143649007923380206012614291248818875408460363608876601967125297555327
0664935503121344668679253586680701811726530087202384230414911810863618314033646652
2061206141951185238106894785222904585003771240585177359575549048733645110060764934
9411988370311457062183683738274960619964670040951089819109329724438803181249860372
9149819276818996080150413480956517439479343800901740082371224292672666708058675522
1944042735253760158496604638814506996781407897328936020429140739083247736561885205
0407641614830863239876269085555296901159746478638847340801030935058960557004996416
1396945310599219789500023705427825104694790800466593584712352146509412823018392111
2654394289987597072193250304580083793856239267949285803273629971043359079188855428
5452361959458729016435118919639281408441775633974425308935355420436591434513397010

```
2899694898788546337574970364913021872121606987932694288914327979921648087090812750
4416392059171329646695727414011305902422623778202510515319635366850294368584171371
3493191592824297269451053166797790253657517558440317438878021487760841211088193210
6355106996703316595072784532204897032930116329260144009780566878025600993935144595
6788799056725116531575428505127716299249591036703294115232791226507371623310110977
6126651582971983013027377622853838089503227323067077686368613498280160646842770552
0255977110211131343453801194967190304682190995624726771490647136246880633201470138
6112366647713223186539640845234156055205801599057535580086545347419203607353058039
4853579519359635514732732936139024053080365344530125639926262555278027296328773095
8159782953796513150995464743526656074041231903855679005521859654697003788940019969
4097745695691796081116975822469712059470024257703207123760996363342953271050007476
8760997766929993295123184390860480823644329656940979911860918947182824769930053562
5210901697950300611563999494279098669625352630883895284076394212895323814714556807
6992367944029018547464932414628843179588958088519628948333089297891094710225091820
1172898433827507402794810689426726617211542897780459542267628307298999861544354457
7108870214559672594945582955906351040354426013574942127371237185878276121379144029
9417792513962880202726327544058990807884871136657965847594284983975972518363508589
7147143102008936398341539810045877012243040080552986250008757577643711986320479317
3906271023007297687515426437502455486892740273605886650280334721687663098106657479
3201396079105280448978225007142675659676090952401166156192762058676863349381051705
4301164195917985268434799501486383755577723597721950719791800581683611588936334765
8385339637837023208818060143429242502362550954776616591843135098247066190158474795
2153423916872344375641603766972834312179049040611160209093197898132950854183653001
9119873096060761023463689715645176680343302772610251742799944016803026455934621906
7240694697187357348627784198313506452113113253421393470444999891789180408898611540
3780434311950357656787764852420748204913306594841919774528186310074650041911140958
2004677828199192922314607237007742031764178371838082393569045205321492201380881782
4519231623673246743481883945343004857732094154811800318581908811726260098825407473
0685489980043709520662450406196004407737118730998156418876361965060647187938244801
0838573103947025904805600341281274836590172125624107399103454285353587092410655695
4550630883185774183163293897930334035099213825110705239552276005051940545633369305
9424601894323299920562098043657864989517685084192520140115744690794968493713722662
7678260772057135051442435037045621661826828560589632952600904660998256395760050804
2036595609231175140351241896447680554741359893239069670477590181301929717579755471
0255903345883978921704509725942884441245410952436632602704933155023825251904296457
0060557214920658995519278612341056639440267671705105854239969899970113180667788376
8436372765140174571578232399037004857441511555842741840811265915962363233893562197
6508177543944576986503224805851014013059103091144093863830417353139695186205757298
7344745529698112205735650766931917417560562925910566301326768711115732730240384522
2106340723540770127109054532811497983971609150975822296508132068661821103765239779
1550913501805518053866451396879941495712888855534086713741921405427824114255892768
3720882708658233026834379453555158668692552849096698203402596780541731193503359229
9655810685697101630893110669511651313389857179278900163451260454639093646004485423
2189543902411265107725237068771892798571647019598800541803284940785035333331470941
4794667351158987353386776573290567239065709230637745804705789036115229996878264601
3083978062";

    cout << "\nmensaje: \n" << message << endl;

    cout << "\nmensaje decifrado: " << endl;
    cout << Elle.decrypt(message) << endl;*/

    /*ZZ n=
conv<ZZ>("853991922719896096503152601516856098693891821176223242685212919844208112
8606266812965474423514260301370452733540760560658962552656301537875246168230782105
```

```
2579684275448776212306112922545681109178618798090130834664457261301431717909117641
4240666481797844358448664672963675776535072548378114898923212457412662294522440252
7862321175537556593241910726554277889051590319896721594908297650416866781267903004
4230054587815089927460923550663681303107337679126994703508166888754057212216526773
3614789006301134608595826246786118625402728175031177671563655976662151785830751312
85929109775716112573191906301872904545989576229021 71");
   ZZ e=
conv<ZZ>("8355572095329378239614449666121445363109333394522945184351360969915327 76
2283885135927417629864524571641058079948221737692176994591249324455944905035840775
2924921797079034310359090205882622610610193614608737352128716359413459308157267751
4875679754012793932079852943665384001515007597515170415593053317733276 01");

   string firma =
"0133707929286786546272813985966049592790252479368963963842346063740105199527127 55
3880389242335140489950994729070620566667513397358600117079279076501143160853099267
0089855600095241556849467581874898147176415592241961980255278976254715788516636912
5974093955115194216885896318925781867239739737195823540084845729975065931512990538
4982950609042953615233430146803237145852544881166617002059490265334080334979593578
4479344646774651717143767801055750055643232155635333642258199267581851043903510834
8606123094867694614303214940971635340718730123916177129401058378745407978509592864
2197697265753154592998833084746036004352484302598941690137816474219242302085339122
6584586519410935060137709999884902563272543231836792533276251231318515286872040095
3929212369614154249979818023143892227214082829168331025547808944724431251785124073
2444922392761730906454202881917678055140942103094107336344693056132712409426444003
6213516235540573042448226494173995242666764649604000765555773914019338675257165506
8332145397932430996507881357625323177857702250273048339487478211592172568543394528
5307548286657766056719992392127917328754581122544879466770546154778594201881939057
6152287954303694403703734163510782747879787141284277511066686239924568348907301233
27564";
   cout << "\nfirma: \n" << firma << endl;
   cout << "\nfirma: " << endl;
   cout << Elle.remove_sign(firma,n,e);*/


   return 0;
}
```

2. Generación de claves

o Generación de números aleatorios

// En esta función accedemos a la arquitectura del computador donde en la variable processInfo guardamos los procesos:

```
void random::fillWithMemoryInfo() {
   DWORD aProcesses[1024], cbNeeded, cProcesses;
   if (EnumProcesses(aProcesses, sizeof(aProcesses), &cbNeeded)) {//obtiene los
procesos en aProcesses
      cProcesses = cbNeeded / sizeof(DWORD);
```

//Calcula cuántos procesos fueron retornados.

```
        for (int i = 100; i < cProcesses; i++){
            HANDLE hProcess = OpenProcess(PROCESS_QUERY_INFORMATION |
PROCESS_VM_READ, FALSE, aProcesses[i]);
            if (NULL != hProcess){
                PROCESS_MEMORY_COUNTERS pmc;
                if (GetProcessMemoryInfo(hProcess, &pmc, sizeof(pmc))) {

processInfo+=(to_string(pmc.PageFaultCount)+to_string(pmc.WorkingSetSize)+

to_string(pmc.QuotaPagedPoolUsage)+to_string(pmc.QuotaNonPagedPoolUsage)+to
_string(pmc.PeakPagefileUsage));
                }
            }
            CloseHandle(hProcess);
        }
    }
}
//Generamos nuestra semilla
vector<int> random::generateSeed(){
    if((processInfo.size()<contador+15)||processInfo.empty()){
        processInfo.clear();
        fillWithMemoryInfo();
        contador=0;
    }
    vector<int> k;
    for(int i=0, j=contador;i<5;i++,j+=3){
        int n=stoi(processInfo.substr(j,3));
        while(n>255) n>>=1;
        while(n<128) n<<=1;
        k.push_back(n);
    }
    contador+=15;
    return k;
}

vector<bool> random::RC4(vector<int> semilla){
    vector<int> Or;
    {//permite que S solo pertenezca a este scope
        vector<int> S;
        for(int i=0;i<256;i++) S.push_back(i);
        {//permite que k solo exista en este scope, optimiza memoria
            vector<int> K;
            for(int i=0,k=0;i<=51;i++)
                for(int j=0;j<5;j++,k++)
                    K.push_back(semilla[j]);
            for(int i=0,f=0;i<256;i++){
                f=modint(f+S[i]+K[i], 256);
                swap(S[i],S[f]);
            }
        }
        for(int i=0,f=0,k=0;k<8;k++){
            i=modint(i + 1, 256);
            f=modint(f+S.at(i), 256);
            swap(S.at(i),S.at(f));
            Or.push_back(S.at(modint(S.at(i) + S.at(f), 256)));//t
```

```
        }
    }
    vector<bool> out;
    for(int i=0;i<8;i++){
        bitset<8> aux(Or[i]);
        for(int j=0;j<8;j++)
            out.push_back(aux[j]);
    }
    return out;
}

void izqRotate(vector<bool> &vec, int times){
    //Rota times veces el vector llevando el primero al ultimo y borrando los primeros
    vec.insert(vec.end(),vec.begin(),vec.begin()+times);//Agrega al final del vector
times elementos del inicio del vector
    vec.erase(vec.begin(),vec.begin()+times);//borra los primeros times valores.
}


//Arrays que tienen maximo como número máximo 64 y son al azar
//PC_1 tiene 56 números y PC_2 tiene 48
int
PC_1[]={56,48,40,32,24,16,8,0,57,49,41,33,25,17,9,1,58,50,42,34,26,18,10,2,59,51,4
3,35,62,54,46,38,30,22,14,6,61,53,45,37,29,21,13,5,60,52,44,36,28,20,12,4,27,19,11,
3};
int
PC_2[]={13,16,10,23,0,4,2,27,14,5,20,9,22,18,11,3,25,7,15,6,26,19,12,1,40,51,30,36,
46,54,29,39,50,44,32,47,43,48,38,55,33,52,45,41,49,35,28,31};
int rotaciones[]={1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1,1,1,2,2,2,2};//propia del DES
//PC elección permutada
//Hacemos uso de la generación de claves de DES
//Donde teóricamente se usan 56 bits de la clave
//Teóricamente son 64 bits del PC_1 pero por paridad que no la necesitamos usamos
sólo 56 bits
//Teóricamente se usan 48 bits de la subclave para PC_2

vector<bool> DES(int bits=1024){
    vector<bool> K;//para guardar todas las k
    {
    vector<bool>k=RC4(generateSeed());//64 bools o bits
    int nBits=bits/48+1;
    vector<bool> c; vector<bool> d;
    for(int i=0;i<28;i++) c.push_back(k.at(PC_1[i]));//Añade cierta posición de k en c
entre las posiciones adquiridas esta los primeros 28 números de PC_1
    for(int i=28;i<56;i++) d.push_back(k.at(PC_1[i]));//Añade cierta posición de k en d
entre las posiciones adquiridas está de 28 a 56 números de PC_1
    for(int i=0;i<nBits;i++){
        izqRotate(c,rotaciones[i]);
        izqRotate(d,rotaciones[i]);//Rotara los vectores rotaciones[i] veces
        vector<bool> both(c.begin(),c.end());//Vector duplicamos con los valores de c
        both.insert(both.end(),d.begin(),d.end());//Agregamos todo el vector de d
        for(int i=0;i<48;i++) K.push_back(both.at(PC_2[i]));//Agregamos a k cierta
posición de both entre las posiciones 0-48 de Pc_2
    }
    }
```

```
        K.resize(bits);//Modifica el k para quedar con cierto número de bits
        K[0]=1; K[bits-1]=1;//Primer y ultimo valor de k sera 1
        return K;
    }

    //Convierte un vector booleano a ZZ tomándolo como número binario
    ZZ random::generate_random(int bits){
        return Random_Number(DES(bits));
    }
```

o Generación de primos

```
    bool Miller_Rabin(ZZ d, ZZ n){

        ZZ a; a=2;
        ZZ x = Power_mod(a, d, n);
        if (x == 1  || x == n-1)return true;
        while (d != n-1){

            x = modulo((x * x), n);
            d *= 2;
            if (x == 1)return false;
            if (x == n-1)return true;
        }

        return false;
    }
    bool isPrime(ZZ n){
        ZZ k; k=0;
        if (n <= 1 or n == 4)return false;
        if (n <= 3)return true;

        ZZ d = n - 1;
        while (even(d)){
            //d /= 2;
            d >>= 1;
            k++;
        }
        for (int i = 0; i < k+1; i++)
            if (!Miller_Rabin(d, n))
                return false;

        return true;
    }

    ZZ gen_prime(ZZ n){//Encuentra el primo más cercano a un número aleatorio
        ZZ a= modulo(n, ZZ(6));
        n-=a+ZZ(5); //para obtener la forma 6n+5
        //Los números primos tienen la forma de 6n+1 y  6n+5, no necesariamente todos
    los números obtenidos con esa
        if(isPrime(n)) return n;//Si es primo retorna ese valor
        else{//Caso contrario busca los siguientes números de la forma 6n+1 y  6n+5
    hasta encontrar un primo
            while(true){
```

```
            n+=2;
            if(isPrime(n)) return n;
            n+=4;
            if(isPrime(n)) return n;
        }
    }
}
```

o Inversa

```
//evalúa si mcd es 1
bool Existe_Inversa(ZZ x, ZZ y){
    if(even(x)&&even(y)) return 0;
    while(x!=0){
        while(even(x)) x=x>>1;
        while(even(y)) y=y>>1;
        ZZ t=valAbs((x-y))>>1;
        if(x>=y) x=t;
        else y=t;
    }
    return y==ZZ(1);
}
//Determina si es par o impar
bool even(ZZ a){
    ZZ r=(a>>1)<<1;
    if(r<0) r=ZZ(2)+r;
    return r==a;
}
//Convertir en  positivo-Valor Absoluto
ZZ valAbs(ZZ a){
    if (a<0) return (a*-1);
    return a;
}
```

o Euclides extendido

```
//Implementamos para Algoritmo Binario del MCD y lo combinamos con el
extendido de euclides para retornar un bool para saber si tiene inversa o no.
ZZ inversa(ZZ r1, ZZ r2){
    ZZ s1=ZZ(1), s2=ZZ(0), b=ZZ(r2);
    if (Existe_Inversa(r1, r2)){
        while (r2>0){
            ZZ q=r1/r2;
            ZZ r=r1-q*r2;
            r1=r2;
            r2=r;
            ZZ s=s1-q*s2;
            s1=s2;
            s2=s;
        }
        if(s1<0) return s1+=b;
        return s1;
```

```
    }
    else cout<<"No tiene inversa"<<endl;
    return ZZ(0);
}
//evalúa si mcd es 1
bool Existe_Inversa(ZZ x, ZZ y){
    if(even(x)&&even(y)) return 0;
    while(x!=0){
        while(even(x)) x=x>>1;
        while(even(y)) y=y>>1;
        ZZ t=valAbs((x-y))>>1;
        if(x>=y) x=t;
        else y=t;
    }
    return y==ZZ(1);
}
//Determina si es par o impar
bool even(ZZ a){
    ZZ r=(a>>1)<<1;
    if(r<0) r=ZZ(2)+r;
    return r==a;
}
//Convertir en  positivo-Valor Absoluto
ZZ valAbs(ZZ a){
    if (a<0) return (a*-1);
    return a;
}
```

3. Formación de Bloques

o Llenar ceros

```
string rsa_bloques::completarCeros(string mensaje,ZZ Nr ){
    int digit =ZZtoStr(Nr).size()-1;
    int c = modint(mensaje.size(),digit);//las letras que no llegan a completar un
bloque
    string cero(digit-c,'0');///la cantidad de 0s que faltan
    return cero+mensaje;
}
int modint(int a,int n){
    int q=a/n;
    if(q<0) q--;
    return a-(n*q);
}
```

o ZZ a string

```
// Convertimos el número en string lo utilizamos en el cifrado y descifrado con y sin
firma digital.

string rsa_bloques::ZZtoStr(ZZ z){
    stringstream ss;
    ss<<z;
```

```
    return ss.str();
}
```

o Dividir bloques

```
string rsa_bloques::dividirBloques(string mensaje){
    //Divide en bloques y completa para tener dígitos al igual que el número de dígitos mayor
    int digit=to_string(alfabeto.size()).size();//número de dígitos del alfabeto
    string str;
    {
        string cero(digit,'0');//string hecha de 0 para completar
        for(int i=0;i<mensaje.size();i++){
            size_t pos=alfabeto.find(mensaje.at(i));//size t adquiere cualquier tipo de variable en los vectores
            int len=to_string(pos).size();//la longitud del número
            if(len<digit) str+=cero.substr(0,digit-len);//completa los 0
            str+=to_string(pos);
        }
    }
    digit=ZZtoStr(n).size()-1;
    while(modint(str.size(),digit)) str+="22";//Aquí completamos el mensaje con W para evitar espacios en blanco
    return str;
}
```

4. Exponenciación modular

```
ZZ Power_mod(ZZ base, ZZ exp, ZZ n){
    //Usamos exponenciación modular el documento
    //Exponenciación Modular Binaria
    ZZ salida(1);
    do{
        if(!even(exp))
            salida= modulo(salida * base, n);
        base= modulo(base * base, n);
        exp>>=1;
    }while(exp!=ZZ(0));
    return salida;
}
ZZ modulo(ZZ a, ZZ b){
    ZZ q= a / b;
    if(q<0) q--;
    return a-(b * q);
}
```

5. Función de cifrado

```
string rsa_bloques::encrypt(string mensaje, ZZ Nr, ZZ Er){
    int digit=ZZtoStr(Nr).size()-1;
    string out;
```

```
    for(int i=0; i < mensaje.size(); i+=digit){
        ZZ p(conv<ZZ>(mensaje.substr(i, digit).c_str()));
        p=Power_mod(p,Er,Nr);
        string ceros((digit+1-ZZtoStr(p).size()),'0');
        out+=ceros+ZZtoStr(p);
    }
    return out;
}
```

6. Función de descifrado

```
string rsa_bloques::decrypt(string mensaje){
    string salida;
    int digitN=ZZtoStr(n).size();
    int digit=digitN-1;
    for(int i=0;i<mensaje.size();i+=digitN){
        ZZ c(conv<ZZ>(mensaje.substr(i,digitN).c_str()));
        c=TRC(c);
        string ceros((digit-ZZtoStr(c).size()),'0');
        salida+=ceros+ZZtoStr(c);
    }
    digit=to_string(alfabeto.size()).size();
    string outLetters;
    for(int i=0;i<salida.size();i+=digit){
        outLetters+=alfabeto.at(stoi(salida.substr(i,digit)));
    }
    return outLetters;
}
```

7. Firma digital

- Cifrado:

```
string rsa_bloques::sign_encrypt(string mensajeL, ZZ Nr, ZZ Er){
    string aux = encryptD(mensajeL);
    return encrypt(completarCeros(aux, Nr), Nr, Er);
}


string rsa_bloques::encryptD(string mensajeL){
    string mensajeNo = dividirBloques(mensajeL);
    int digit=ZZtoStr(n).size()-1;
    string out;
    for(int i=0;i<mensajeNo.size();i+=digit){
        ZZ c(conv<ZZ>(mensajeNo.substr(i,digit).c_str()));
        c=TRC(c);//se realiza una segunda versión para optimizar con el TRC y mayor
velocidad
        string ceros((digit+1-ZZtoStr(c).size()),'0');
```

```
        out=out+ceros+ZZtoStr(c);
    }
    return out;
}
```

- Descifrado:

```
string rsa_bloques::remove_sign(string mensaje, ZZ Ne, ZZ Ee){
    return decryptE(mensaje, Ne, Ee);
}


string rsa_bloques::decryptE(string mensaje, ZZ Ne, ZZ Ee){
    string salida;
    int digitN=ZZtoStr(n).size();
    int digit=digitN-1;
    for(int i=0;i<mensaje.size();i+=digitN){
        ZZ c(conv<ZZ>(mensaje.substr(i,digitN).c_str()));
        c=TRC(c);
        string ceros((digit-ZZtoStr(c).size()),'0');
        salida+=ceros+ZZtoStr(c);
    }
    string mensajeNo = salida;
    int a= modint(mensajeNo.size(), ZZtoStr(Ne).size());
    mensajeNo = mensajeNo.substr(a);

    string salida2;
    digit=ZZtoStr(Ne).size()-1;
    digitN=digit+1;
    for(int i=0; i < mensajeNo.size(); i+=digitN){
        ZZ c(conv<ZZ>(mensajeNo.substr(i, digitN).c_str()));
        c=Power_mod(c,Ee,Ne);
        string ceros((digit-ZZtoStr(c).size()),'0');
        salida2+=ceros+ZZtoStr(c);
    }
    digit=to_string(alfabeto.size()).size();
    string outLetters;
    for(int i=0;i<salida2.size();i+=digit){
        outLetters+=alfabeto.at(stoi(salida2.substr(i,digit)));
    }
    return outLetters;
}
```

8. TRC

```
ZZ rsa_bloques::TRC(ZZ M){
    ZZ q1= inversa(modulo(q, p), p);
    ZZ a1= Power_mod(modulo(M, p), modulo(d, p - 1), p);
    ZZ q2= inversa(modulo(p, q), q);
    ZZ a2= Power_mod(modulo(M, q), modulo(d, q - 1), q);
    return modulo(modulo(a1 * q * q1, n) + modulo(a2 * p * q2, n), n);
}
```

```
ZZ Power_mod(ZZ base, ZZ exp, ZZ n){
    //Usamos exponenciación modular el documento
    //Exponenciación Modular Binaria
    ZZ salida(1);
    do{
        if(!even(exp))
            salida= modulo(salida * base, n);
        base= modulo(base * base, n);
        exp>>=1;
    }while(exp!=ZZ(0));
    return salida;
}
```