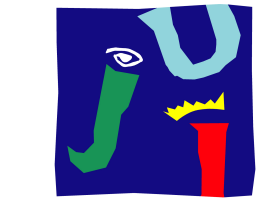

UNIVERSITAT JAUME·I

Màster Universitari en Matemàtica Computacional

Treball final de màster

Una introducció al sistema d'encriptació RSA

Autora:
Marta Iborra Clari

Tutor:
Carlos Galindo Pastor

Data de lectura: octubre del 2023
Curs acadèmic 2022/2023

Resum

Al llarg de la història s'han utilitzat diferents sistemes d'encriptació segons el context per tal de protegir el tràfic de dades a través d'un canal no segur. Els primers sistemes d'encriptació a descobrir-se foren els de clau privada, en els quals la clau és compartida per les parts implicades en la comunicació per a xifrar i desxifrar missatges. El més usat avui en dia és l'AES. Més tard, sorgiren els sistemes de clau pública. En aquests, hi ha dues claus: una pública accessible per a tothom que permet encriptar dades, i una privada que posseeix només el destinatari de les dades que permet desxifrar-les. En molts d'aquests sistemes l'aritmètica modular és essencial, i en el cas del sistema més usat de clau pública (RSA) l'estudi de la factorització d'enters és la base de la seva seguretat. De fet, per tal que un sistema siga bo, cal que aquest sistema siga factible d'aplicar amb els recursos del moment, gens factible de desxifrar sense la clau privada i resistent front als diferents atacs que es puguen realitzar. D'altra banda, hi ha un altre tipus de xifratge (l'homomòrfic) que permet poder operar amb dades xifrades de manera que el servidor pot realitzar les operacions sense tindre accés a les dades originals ni saber-ne el resultat i és només el posseïdor de les dades qui té accés a aquestes. Malgrat el seu potencial per a aplicacions al núvol, aquest xifratge no compleix en l'actualitat la característica de ser factible d'aplicar amb els recursos que hom pot tenir a l'abast i encara està en fase d'investigació.

Paraules clau

Sistemes d'encriptació. RSA.

Keywords

Cryptosystems. RSA.

Índex

Capítol 1

Introducció

La necessitat de poder tindre una comunicació segura a través d'un canal no segur està present des de fa molts segles. La ciència que s'encarrega d'estudiar-ho rep el nom de criptologia, i estudia tant les maneres que hi ha de xifrar un missatge (criptografia) com les possibles maneres que té un oponent d'obtindre el missatge original a partir del xifrat per a cada mètode en concret (criptoanàlisi).

Es considera que una de les primeres maneres de xifrar un missatge (sistema d'encriptació) va ser establerta per Juli Cèsar. Aquest usava el seu sistema d'encriptació per a comunicar les estratègies a seguir en el camp de batalla. Així, en cas que algú interferira el missatge, no podia saber l'estratègia que seguirien els romans. Encara avui, el govern militar usa l'encriptació per a protegir els seus avenços i evitar que la informació que té el govern siga usada en contra seva. No obstant això, actualment s'usa en altres contextos molt diferents com són les comunicacions quotidianes a través d'Internet per tal d'oferir privacitat. Segons el context de comunicació, convé aplicar un sistema d'encriptació de clau privada, en el qual l'emissor i el receptor comparteixen la mateixa clau per a xifrar i desxifrar els missatges, o un sistema de clau pública, en el qual qualsevol pot xifrar un missatge a un destinatari amb la clau pública, però només el destinatari pot desencriptar el missatge amb la clau privada. El sistema de clau privada sol ser més ràpid i el més usat avui en dia és l'AES. Pel que fa a la clau pública, aquest sistema és més segur i els més usats en l'actualitat són el RSA i el logarítmic.

En aquest treball, els capítols 2, 3, 4 i 5 tracten de donar la base matemàtica per a endinsar-nos en el món de la criptologia i més concretament, en el RSA. I, al capítol 6 és on ens endinse'm en aquest món. En aquest últim capítol es presenta el sistema

d'encriptació de clau privada junt amb uns exemples i un anàlisi de la seva seguretat. Seguidament es tracta el sistema de clau pública i s'analitza el RSA amb deteniment. Finalment, es fa una breu introducció del xifratge homomòrfic, un xifratge que permet realitzar operacions amb dades ja xifrades. Així, un servidor pot realitzar operacions amb aquestes, però és només el propietari de les dades qui pot tindre accés a les dades i al resultat de les operacions. Per acabar, hi ha un últim capítol amb les conclusions que s'han extret d'aquest treball.

Capítol 2

Estimació del temps de computació d'operacions aritmètiques

Malgrat que cada cop es poden construir ordinadors més eficients, capaços de realitzar més operacions amb menys temps, aquesta eficiència sempre té un límit al qual hom pot arribar a topar-se segons els treballs que es realitzen. És per això que a l'hora de programar és essencial saber el temps que un computador requereix per a realitzar una operació segons el tipus d'aquesta per tal d'aprofitar al màxim els recursos. Així doncs, interessa estudiar el temps des del punt de vista informàtic i no humà, és a dir, el temps d'execució d'operacions amb el sistema binari.

És sabut per la gran part de la societat que així com el sistema numèric més estès utilitzat per l'ésser humà és el decimal (que treballa amb potències de 10), el sistema numèric que usen internament les màquines sol ser el binari. Així, el nombre 201 es pot reconstruir en el sistema decimal (o base 10) com a $201 = 2 \cdot 10^2 + 0 \cdot 10^1 + 1 \cdot 10^0$. En aquesta representació, cadascun dels nombres que multiplica una potència de 10 és menor estricte que 10. Per a representar aquest mateix nombre en el sistema binari (o base 2), s'expressa aquest mateix nombre com a suma de factors enters entre 0 i 1 multiplicant potències de 2 i la seua representació és la consecució dels factors que multipliquen cada potència. És a dir, $201 = 1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = (11001001)_2$.

2.1 Operacions en el sistema binari

A l'hora d'operar dos nombres representats en el sistema binari, el procediment a seguir és el mateix que per al sistema decimal. Per exemple, si es desitja dividir $(11001001)_2$ entre $(100111)_2$, el procediment seria el següent:

$$\begin{array}{r|l} 11001001 & 100111 \\ \hline -100111 & 101 \\ \hline 101101 & \\ -100111 & \\ \hline 110 & \end{array}$$

Per a la representació aproximada de nombres decimals, com per exemple el nombre π, s'usen tant potències positives, com negatives. Així,

$$\begin{aligned} \pi &\approx 1 \cdot 2 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 0 \cdot 2^{-4} + 0 \cdot 2^{-5} + 1 \cdot 2^{-6} \\ &\approx (11.001001)_2. \end{aligned}$$

Si s'ha seguit el procediment de fer operacions amb diferents bases, s'haurà pogut observar que la quantitat d'operacions ve determinada per la quantitat de xifres que s'usen per representar els operands. Per a un nombre enter n i una base b, si $b^{k-1} \leq n < b^k$, n es representa amb k xifres (o dígits) en la base donada. Per la definició dels logaritmes, podem obtindre la següent expressió per a determinar k:

$$k = \text{ Quantitat de xifres } = [\log_b n] + 1 = \left[\frac{\log n}{\log b}\right] + 1.$$

A partir d'aquí, log fa referència al logaritme natural $\log_e$ i $[x]$ el major nombre enter menor o igual que x.

A continuació es determinarà per a cada operació el temps en base 2 per ser la corresponent al sistema binari.

Addició

Donats dos nombres de la mateixa longitud k en base 2 (en cas de no ser així s'ompliria de zeros a l'esquerra el nombre de menor longitud fins tindre la mateixa longitud que l'altre), per tal de sumar-los hauríem de repetir k vegades els següents passos:

1. Mirar la xifra que ocupa la mateixa posició d'ambdós operands i també si en portem o no.

2. Si ambdós bits són 0 i no en portem, aleshores el dígit corresponent a eixa posició del resultat serà un 0. Seguidament, tornem al pas 1 amb la posició que està a l'esquerra de l'actual.

3. Si o bé, ambdós bits són 0 i en portem una, o bé, un bit és 0, l'altre 1 i no en portem cap, llavors el resultat en aquesta posició serà 1. Seguidament, tornem al pas 1 amb la posició que està a l'esquerra de l'actual.

4. Si un dels bits és 0, l'altre 1 i en portem una, o si ambdós són 1 i no en portem cap, el resultat en eixa posició serà un 0. A continuació, tornem al pas 1 amb la posició que està a l'esquerra de l'actual portant-ne una.

5. Si els dos bits són 1 i en portem una, el resultat en eixa posició serà 1. Seguidament, tornem al pas 1 amb la posició que està a l'esquerra de l'actual portant-ne una.

Fer aquest procediment per a una posició determinada s'anomena operació bit. Si els nombres són de longitud k es realitzaran k operacions bit.

Producte

Donats dos enters binaris n i m de longitud k i l respectivament, vegem el mètode habitual per a multiplicar. Per exemple:

				1	1	1	0	1
					1	1	0	1
				1	1	1	0	1
		1	1	1	0	1	0	
	1	1	1	0	1			
1	0	1	1	1	1	0	0	1

Després de multiplicar n i m, obtenim un màxim de l files (una fila menys per cada bit de m que siga 0), on cada fila consisteix en una còpia de n desplaçada cap a l'esquerra certes posicions. Suposem que hi ha $l' \leq l$ files. Com que una operació bit no correspon a la suma d'una columna, sinó que correspon a la suma de dos elements, hem de realitzar les sumes fila per fila acumulant-ne el resultat. A cada etapa, observem quantes xifres a l'esquerra s'ha desplaçat n per a formar la nova fila. Copiem els bits que estan més a la dreta de la suma parcial i sumen a n la resta de la suma parcial (per a realitzar aquesta suma calen k operacions bit). Així, la multiplicació es pot fer en $l' - 1$ sumes, cadascuna requerint k operacions bit.

$$\text{Temps(multiplicar } n \text{ de longitud } k \text{ per } m \text{ de longitud } l) < kl.$$

Cal remarcar que en aquest càlcul s'han menyspreat tant el temps de desplaçament de xifres com el de còpia per ser en la pràctica un procediment més ràpid.

Si recordem la fórmula anterior amb logaritmes, es podria escriure l'estimació d'aquesta manera:

$$k = [\log_2 n] + 1 \leq \frac{\log n}{\log 2} + 1 \text{ i } l = [\log_2 m] + 1 \leq \frac{\log m}{\log 2} + 1 \ .$$

Aleshores obtenim:

$$\text{Temps(multiplicar } n \text{ de longitud } k \text{ per } m \text{ de longitud } l) < kl$$
$$< \frac{\log n \cdot m}{\log^2 2} + \frac{\log n}{\log 2} + \frac{\log m}{\log 2} + 1.$$

L'estimació per a les operacions de resta i divisió és la mateixa que per a les operacions d'addició i multiplicació respectivament:

Temps(restar k-bit de l-bit) $\leq \max(k, l)$.

Temps(dividir k bit entre l-bit) $\leq kl$.

Nota 1. Com a exemple de com aproximar el nombre d'operacions que necessita un ordinador per a fer un càlcul matemàtic, indiquem com trobar un fita superior del nombre d'operacions bit requerides per a multiplicar un polinomi $\sum a_i x^i$ de grau $\leq n_1$ amb un polinomi $\sum b_j x^j$ de grau $\leq n_2$ els coeficients dels quals són enters positius $\leq m$ (suposem que $n_2 \leq n_1$).

Per a això, observem que cada coeficient de x^ν en el producte (on $0 \leq \nu \leq n_1 + n_2$) ve donat per $\sum_{i+j=\nu} a_i b_j$. Aquest càlcul requereix de com a molt $n_2 + 1$ multiplicacions i n_2 sumes.

Els operands de les multiplicacions estan fitats per m, i els de les sumes per m^2. Com que cal calcular la suma parcial de com a molt n_2 nombres, però, la fita dels operands de la suma serà $n_2 m^2$. Així, la quantitat d'operacions bit requerides per a obtindre el coeficient de x^ν en el pitjor dels escenaris és

$$(n_2 + 1)\left(\log_2 m + 1\right)^2 + n_2\left(\log_2\left(n_2 m^2\right) + 1\right).$$

Com que hi ha $n_1 + n_2 + 1$ valors de ν, l'estimació del temps de multiplicació dels dos polinomis és

$$(n_1 + n_2 + 1)\left((n_2 + 1)\left(\log_2 m + 1\right)^2 + n_2\left(\log_2\left(n_2 m^2\right) + 1\right)\right).$$

Si ignorem els 1, es pot obtindre la següent estimació més compacta però menys rigorosa

$$\frac{n_2\,(n_1+n_2)}{\log 2}\left(\frac{(\log m)^2}{\log 2}+(\log n_2+2\log m)\right).$$

2.2 Notació O majúscula

Per tal de tindre una manera de determinar el cost computacional d'un algoritme de manera general i, en definitiva, poder comparar uns algoritmes amb altres s'usa la notació O majúscula. Aquesta notació que es defineix a continuació fou inventada pels matemàtics Paul Bachmann [5] i Edmund Landau.

Definició 1. *Siguen $f\,(n_1,n_2,\ldots,n_r)$ i $g\,(n_1,n_2,\ldots,n_r)$ dues funcions els dominis de les quals són subconjunts de totes les r-tuples d'enters positius. Suposem que existeixen unes constants B i C tal que quan $n_j > B$, per a tot j ambdues funcions són positives, i $f\,(n_1,n_2,\ldots,n_r) < Cg\,(n_1,n_2,\ldots,n_r)$. Aleshores, es diu que f està fitada per g i es denota per $f = O(g)$.*

Com a exemple, indiquem com usar la O majúscula en dos casos.

1. Siga $f(n)$ un polinomi qualsevol de grau d amb coeficient líder positiu. Aleshores $f(n) = O\left(n^d\right)$.
2. El resultat de la nota anterior el podríem reescriure com a:

$$\begin{aligned}\text{Temps}\left(\sum a_i x^i \cdot \sum b_j x^j\right) &= O\left(\frac{n_2(n_1+n_2)}{\log 2}\left(\frac{(\log m)^2}{\log 2}+\log n_2+2\log m\right)\right)\\ &= O\left(n_2 n_1\left(\log m(\log m+2)+\log n_2\right)\right)\\ &= O\left(n_1 n_2\left((\log m)^2+\log n_2\right)\right).\end{aligned}$$

D'aquest últim exemple es pot observar que, per definició de la notació O majúscula, les constants cauen i expressions amb potències es redueixen a l'expressió elevada a la major potència que apareix.

La següent definició introdueix aquells algoritmes que permeten fer càlculs de manera eficient amb nombres grans.

Definició 2. *Un algoritme per a realitzar un càlcul amb els enters $n_1, n_2, \ldots, n_r$ de $k_1, k_2, \ldots, k_r$ bits, respectivament, es diu que és de temps polinòmic si existeixen enters $d_1, d_2, \ldots, d_r$ tal que el nombre d'operacions bit requerides per a aplicar l'algoritme és $O\left(k_1^{d_1} k_2^{d_2} \cdots k_r^{d_r}\right)$.*

Exemples d'algoritmes de temps polinòmic són les operacions aritmètiques usuals $(+, -, \times, \div)$. No tots però, tenen perquè ser polinòmics. Veiem per exemple el cost de calcular una arrel quadrada.

Nota 2. Calcular l'arrel quadrada d'un nombre binari n requereix $O(\log^3 n)$ operacions bit.

En efecte, tal com indica Ujjwal Singh [23], de manera similar al procediment usual per a calcular l'arrel quadrada d'un nombre decimal, l'algoritme per a calcular l'arrel quadrada d'un nombre binari es basa en el següent.

Siga a un nombre en binari i b un dígit (per a aquesta explicació només, denotarem amb $\mid$ a la funció d'unir les xifres d'un nombre amb les de l'altre), tenim que:

$$\begin{aligned}
(a \mid b)^2 &= (2a + b)^2 \\
&= 4a^2 + 4ab + b^2. \\
(a \mid b)^2 - 4a^2 &= (4a + b)b \\
&= (a \mid 00 + b)b \\
&= (a \mid 0b)b. \\
(a \mid b)^2 - a^2 \mid 00 &= (a \mid 0b)b.
\end{aligned}$$

Així doncs, si sabem tots els bits excepte l'últim, obtenim el bit restant 4 vegades el quadrat del que ja tenim i buscant després un bit d tal que $(a \mid 0d)d$ és igual a la diferència.

Com a exemple indicarem com calcular l'arrel quadrada de 10101001 . Per a això, procedirem de la següent manera (veure la següent imatge).

1. Fem parelles de dreta a esquerra.
2. La parella més a l'esquerra és 10, així que cerquem un dígit d el quadrat del qual siga menor o igual que aquest $(d = 1)$.
3. Restem d a 10 i baixem la següent parella a la dreta del residu. Aquest pas correspon a $(a \mid b)^2 - a^2 \mid 00$.

4. Seguidament, intentem buscar el $(1 \mid 0d)d$ més gran que siga menor que 110 ($d = 1$).
5. Així, es repeteixen els passos 3 i 4 fins que ja s'han baixat totes les parelles.

Finalment, el resultat és 1101.

$\sqrt{10\ 10\ 10\ 01}$	1101
1	$101 \cdot 1 = 101$
1 10	$1100 \cdot 0 = 0$
1 01	
0 01 10	$11001 \cdot 1 = 11001$
0	
1 10 01	
1 10 01	
0	

Analitzant les operacions que s'han de fer, observem que a cada parella que baixem realitzem un màxim de dues multiplicacions i una resta. El cost de realitzar una multiplicació és com a màxim $O(\log^2 n)$ (el de la resta no es contempla perquè és menor). Aquest procediment es realitzarà $\log n/2$ vegades. Per tant, ignorant el cost de baixar les xifres a cada pas, el cost de calcular l'arrel quadrada amb aquest algoritme senzill és de $O(\log^3 n)$ -tot i que hi ha algoritmes més eficients que aquest-.

Referències. Per a aquest capítol, s'ha usat com a referència el llibre *A course in number theory and cryptography* de N. Koblitz [20] i *An introduction to the theory of numbers* de G. H. Hardy i E. M. Wright [14].

Capítol 3

Divisibilitat i l'algoritme euclidià

Els nombres primers són fonamentals en la criptologia ja que els algoritmes criptogràfics com RSA es basen en la dificultat de factoritzar nombres grans en els seus factors primers. Per a l'estudi d'aquesta qüestió, un primer pas és l'estudi de la divisibilitat. Començarem amb unes definicions.

Definició 3. *Donats dos enters a i b, es diu que a divideix a b i es denota per $a|b$ si existeix un enter d tal que $b = ad$. En eixe cas es diu que a és un divisor de b. Tot enter $b > 1$ té com a mínim dos divisors: la unitat i ell mateix.*
Un divisor propi és un divisor diferent dels trivial, i.e. diferent de la unitat i de b. Així, un nombre primer és per definició un nombre enter major que 1 que no té cap divisor propi; un nombre es diu compost si té com a mínim un divisor no trivial (diferent de la unitat i de si mateix).

D'aquestes definicions es poden deduir les següents propietats:

1. Si $a \mid b$ i c és un enter qualsevol, aleshores $a \mid bc$.
2. Si $a \mid b$ i $b \mid c$, llavors $a \mid c$.
3. Si $a \mid b$ i $a \mid c$, aleshores $a \mid b \pm c$.

Si p és un nombre primer i α és un enter no negatiu, aleshores s'usa la notació $p^\alpha \| b$ per a indicar que p^α és la potència més gran de p que divideix a b, i.e., que $p^\alpha \mid b$ i $p^{\alpha+1} \nmid b$. En eixe cas, es diu que p^α divideix exactament a b.

Teorema 1 (Teorema fonamental de l'aritmètica)**.** *Tot natural n es pot escriure de manera única com a producte de nombres primers.*

Com a conseqüència d'aquest teorema -enunciat per primera vegada per Abu Hasan Muhammad ibn Hasan [3], tot i que altres l'atribueixen a Gauss [12]-, s'obtenen dues propietats més:

4. Si un nombre primer p divideix ab, aleshores o $p \mid a$ o $p \mid b$.

5. Si $m \mid a$ i $n \mid a$, i si m i n no tenen divisors més gran que la unitat en comú, aleshores $mn \mid a$.

Una altra conseqüència de la factorització única és que dona un mètode sistemàtic per a trobar tots els divisors de n un cop n està escrit com a producte de potències de nombres primers. En concret, qualsevol divisor d de n ha de ser un producte dels mateixos primers elevats a potències menors o iguals que la potència que divideix exactament a n. És a dir, si $p^{\alpha} \| n$, aleshores $p^{\beta} \| d$ per a algun β que satisfà que $0 \leq \beta \leq \alpha$.

Per exemple, si es desitjara conèixer el nombre de divisors de $4200 = 2^3 \cdot 3 \cdot 5^2 \cdot 7$ a partir de la factorització única, resultaria que la quantitat de possibles divisors és el producte del nombre de possibilitats per a cada potència primera. Així, 4200 té $4 \cdot 2 \cdot 3 \cdot 2 = 48$ divisors. La fórmula general diu que un nombre $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_r^{\alpha_r}$ té $(\alpha_1 + 1)(\alpha_2 + 1) \cdots (\alpha_r + 1)$ divisors diferents.

Recordem aquí la definició de màxim comú divisor.

Definició 4. *Donats dos enters a i b, amb un dels dos diferent de 0, el màxim comú divisor de a i b, denotat com a $m.c.d(a, b)$ (o $g.c.d(a, b)$ de l'anglès) és el divisor més gran d que divideix tant a a com a b.*

Una definició equivalent a aquesta és que el m.c.d. és l'enter positiu d que divideix a a i a b i no és divisible per cap altre nombre que divideixi a i b.

Un altre cop, si es té la factorització única d'a i b, es pot obtindre el m.c.d.(a, b) seleccionant els primers que apareixen a ambdós factoritzacions amb la menor potència. Tanmateix, si a o b són nombres enters grans, és força difícil saber la seva factorització en nombres primers. De fet, es tracta d'una àrea de recerca. Per tal d'obtindre el m.c.d. de dos nombres sense necessitat d'obtindre abans la factorització en nombres primers, s'usa l'algoritme euclidià, un algoritme relativament ràpid.

Definició 5 (L'algoritme Euclidià)**.** *Per tal de trobar el m.c.d.*(a, b)*, on* $a > b$*, primer es divideix* a *entre* b *i s'escriu el quocient* q_1 *i el residu* $r_1 : a = q_1 b + r_1$*. Després, es realitza una segona divisió però ara* b *fa el paper d'*a *i* r_1 *el de* b*. Així, dividim* r_1 *entre* b *anotem el quocient* q_2 *i el residu* r_2*. Es continua aquest procediment fins que s'obté un residu que divideix a l'anterior residu; aquest residu diferent de* 0 *és el màxim comú divisor de* a *i* b*.*

Exemple 1. Com a exemple aplicarem l'algoritme euclidià per a trobar el m.c.d.$(1547, 560)$.

$$\begin{aligned} 1547 &= 2 \cdot 560 + 427 \\ 560 &= 1 \cdot 427 + 133 \\ 427 &= 3 \cdot 133 + 28 \\ 133 &= 4 \cdot 28 + 21 \\ 28 &= 1 \cdot 21 + 7. \end{aligned}$$

Com que $7 \mid 21$, m.c.d.$(1547, 560) = 7$.

Aquest algoritme sempre es pot realitzar en un nombre finit de passes tal i com es veurà a continuació.

Proposició 1. *L'algoritme euclidià sempre dona el màxim comú divisor en un nombre finit de passes. A més a més, per a* $a > b$

$$\textit{Temps(Trobar m.c.d. } (a, b) \textit{ amb l'algoritme euclidià)} = O\left(\log^3(a)\right).$$

Demostració. La demostració de la 1a part es basa en que la successió de residus és estrictament decreixent i, per tant, ha d'arribar a 0. Pel que fa al resultat, cal usar la 2a definició de m.c.d. Així, si un nombre divideix a i b, ha de dividir r_1, i com que divideix b i r_1 ha de dividir r_2 i així successivament. D'altra banda, començant de dalt cap a baix tenim que l'últim residu ha de dividir a tots els altres residus anteriors i per tant, també a a i b. Per tant, eixe és el m.c.d., perquè és l'únic nombre que divideix a a i b i al mateix temps és divisible per qualsevol altre nombre que divideix a i b.

A continuació demostrarem l'estimació del temps. Per a fer-ho demostrarem la quantitat de divisions que es fan.

Lema 1: $r_{j+2} < \frac{1}{2} r_j$

Demostració. Si $r_{j+1} \leq \frac{1}{2} r_j$, aleshores tenim que $r_{j+2} < r_{j+1} \leq \frac{1}{2} r_j$. Així doncs, suposem que $r_{j+1} > \frac{1}{2} r_j$. En aquest cas la següent divisió dona $r_j = 1 \cdot r_{j+1} + r_{j+2}$, i per tant $r_{j+2} = r_j - r_{j+1} < \frac{1}{2} r_j$ tal i com es volia demostrar. □

Com que cada dos passos el tamany del residu disminueix com a mínim la meitat i com que aquest mai baixa de 1, hi ha com a màxim $2 \cdot [\log_2 a]$ divisions. En la notació O, açò és $O(\log a)$. Cada divisió implica nombres no més grans de a, així, implica $O\left(\log^2 a\right)$ operacions bit. Per tant, el temps total és $O(\log a) \cdot O\left(\log^2 a\right) = O\left(\log^3 a\right)$. □

Proposició 2. *Siga $d = m.c.d.(a, b)$, on $a > b$. Llavors existeixen enters u i v tal que $d = ua + bv$. És a dir, el m.c.d. de dos nombres es pot expressar com una combinació lineal dels dos nombres amb coeficients enters. A més a més, trobar u i v es pot fer en $O\left(\log^3 a\right)$ operacions bit.*

Demostració. Aplicant l'algoritme d'Euclides s'arriba a la demostració de la proposició. □

Exemple 2. Continuant amb l'exemple anterior, podem expressar 7 com a combinació lineal de 1547 i 560.

$$\begin{aligned}
7 &= 28 - 1 \cdot 21 = 28 - 1(133 - 4 \cdot 28) \\
&= 5 \cdot 28 - 1 \cdot 133 = 5(427 - 3 \cdot 133) - 1 \cdot 133 \\
&= 5 \cdot 427 - 16 \cdot 133 = 5 \cdot 427 - 16(560 - 1 \cdot 427) \\
&= 21 \cdot 427 - 16 \cdot 560 = 21(1547 - 2 \cdot 560) - 16 \cdot 560 \\
&= 21 \cdot 1547 - 58 \cdot 560.
\end{aligned}$$

Definició 6. *Dos nombres a i b es diu que són primers entre si (o coprimers) si el $m.c.d.(a, b) = 1$.*

Introduïm aquí una funció que serà molt útil més endavant.

Definició 7. *Siga n un enter positiu, la funció phi d'Euler, denotada per $\varphi(n)$, és per definició la quantitat de nombres enters no negatius $b < n$ que són primers a n:*

$$\varphi(n) = \mid \{0 \leq b < n \mid \ m.c.d.(b, n) = 1\} \mid,$$

on $\mid A \mid$ denota el cardinal del conjunt A.

Com a conseqüència, $\varphi(1) = 1$ i $\varphi(p) = p - 1$ per a qualsevol nombre primer p. A més a més, es pot demostrar que per a qualsevol potència primera

$$\varphi\left(p^\alpha\right) = p^\alpha - p^{\alpha-1} = p^\alpha \left(1 - \frac{1}{p}\right).$$

Per a demostrar açò, és suficient observar que els nombres de 0 a $p^\alpha - 1$ que no són primers a p^α són precisament els que són divisibles per p, i n'hi ha $p^{\alpha-1}$.

Referències. En aquest capítol s'han usat principalment l'obra de H. Cohen *A course in computational algebraic number theory* [10] i l'obra de N. Koblitz [20].

Capítol 4

Congruències

En aquest capítol s'estudiarà l'aritmètica modular ja que juga un paper important en la criptografia i de fet, és usada en molts algoritmes criptogràfics. La principal eina de l'aritmètica modular és la congruència, que és en realitat una relació binària d'equivalència. És per això que començarem aquest apartat amb la definició d'aquesta última.

Definició 8. *Donats tres enters a, b, m, es diu que a és congruent en b mòdul m i s'escriu $a \equiv b \bmod m$, si la diferència $a-b$ és divisible entre m. El nombre m s'anomena el mòdul de congruència.*

Proposició 3. *Les següents propietats es compleixen.*

1. *(i) $a \equiv a \bmod m$; (ii) $a \equiv b \bmod m$ si i només si $b \equiv a \bmod m$; (iii) si $a \equiv b \bmod m$ i $b \equiv c \bmod m$, aleshores $a \equiv c \bmod m$. Fixat un m, (i) -(iii) impliquen que la congruència mòdul m és una relació d'equivalència.*

2. *Fixat un m, cada classe d'equivalència respecte al mòdul de congruència m té un únic representant entre 0 o $m-1$. El conjunt de classes d'equivalència (anomenat classes residu) es denota per $\mathbb{Z}/\mathrm{m}\mathbb{Z}$. Qualsevol conjunt de representants de les classes residu s'anomena un conjunt complet de residus mòdul m.*

3. *Si $a \equiv b \bmod m$ i $c \equiv d \bmod m$, aleshores $a \pm c \equiv b \pm d \bmod m$ i $ac \equiv bd \bmod m$. Hom diu que el conjunt de classes d'equivalència $\mathbb{Z}/m\mathbb{Z}$ (també denotat per $\mathbb{Z}_m$) és un anell commutatiu.*

4. *Si $a \equiv b \bmod m$, llavors $a \equiv b \bmod d$ per a qualsevol divisor $d \mid m$.*

5. Si $a \equiv b \bmod m, a \equiv b \bmod n$*, i* m *i* n *són coprimers, aleshores* $a \equiv b \bmod mn$*.*

Proposició 4. *Els elements de* $\mathbb{Z}_m$ *que tenen inversos multiplicatius són aquells que són primers a* m*, i.e. els nombres* a *per als quals existeix* b *amb* $ab \equiv 1 \bmod m$ *són precisament els* a *per als quals* $m.c.d(a, m) = 1$*. A més a més, si* $m.c.d.(a, m) = 1$*, aleshores l'invers* b *es pot trobar en* $O\left(\log^3 \mathrm{m}\right)$ *operacions bit.*

El següent exemple és una mostra de com formar inversos en l'anell de congruències.

Exemple 3. Veiem com trobar $160^{-1} \bmod 841$, és a dir, l'invers de 160 mòdul 841.

Cal trobar l'expressió $u \cdot a + v \cdot m = 1$ on $a = 160$ i $m = 841$ en aquest cas. Així, u serà l'invers d'a, perquè $au \equiv 1 \bmod m$.

Mitjançant l'algoritme d'Euclides:

$$\begin{aligned} 841 &= 5 \cdot 160 + 41 \\ 160 &= 3 \cdot 41 + 37 \\ 41 &= 1 \cdot 37 + 4 \\ 37 &= 9 \cdot 4 + 1. \end{aligned}$$

D'on obtenim

$$\begin{aligned} 1 &= 37 - 9 \cdot 4 \\ &= 37 - 9 \cdot (41 - 37) \\ &= -9 \cdot 41 + 10 \cdot 37 \\ &= -9 \cdot 41 + 10 \cdot (160 - 3 \cdot 41) \\ &... \\ 1 &= -39 \cdot 841 + 160 \cdot 205. \end{aligned}$$

Per tant, $160^{-1} \bmod 841 = 205$. Recordem que es tracta d'un dels representants d'una classe d'equivalència.

Indiquem a continuació algunes propietats relacionades amb l'anell de congruències.

Corol·lari 1. *Si* p *és un nombre primer, aleshores tota classe residu diferent de la classe del zero té un invers multiplicatiu que es pot trobar en* $O\left(\log^3 p\right)$ *operacions bit. Aleshores* $\mathbb{Z}/p\mathbb{Z}$ *(o* $\mathbb{F}_p$*) és un cos.*

Corol·lari 2. *Suposem que volem resoldre l'equació lineal de congruència* $ax \equiv b \bmod m$ *on, sense pèrdua de generalitat, assumim que* $0 \leq a$ *i* $b < m$*. Si*

m.c.d. $(a, m) = 1$, *aleshores existeix una solució* x_0 *que es pot trobar en* $O\left(\log^3 m\right)$ *operacions bit, i totes les solucions són de la forma* $x = x_0 + mn$ *on* n *és un enter. D'altra banda, siga* $d = m.c.d.(a, m)$, *aleshores existeix una solució si i només si* $d \mid b$, *i en eixe cas l'equació de congruència és equivalent (i.e. té les mateixes solucions) a la congruència* $a'x \equiv b' \bmod m'$, *on* $a' = a/d, b' = b/d, m' = m/d$.

Corol·lari 3. *Si* $a \equiv b \bmod m$ *i* $c \equiv d \bmod m$, *i si m.c.d* $(c, m) = 1$, *aleshores* $ac^{-1} \equiv bd^{-1} \bmod m$ *(on* c^{-1} *i* d^{-1} *denoten qualsevol enter invers a c i d mòdul* m*).*

Proposició 5 (El petit teorema de Fermat)**.** *Siga* p *un nombre primer. Qualsevol enter satisfà que* $a^p \equiv a \bmod p$, *i qualsevol enter* a *no divisible per* p *satisfà* $a^{p-1} \equiv 1 \bmod p$.

Demostració. Suposem que $p \nmid a$. Primer demostrarem que els enters $0a, 1a, 2a, 3a, \ldots, (p-1)a$ formen un conjunt complet de residus mòdul p. En cas que no ho foren, dos d'ells (ia i ja) haurien d'estar en la mateixa classe residu. És a dir, $ia \equiv ja \bmod p$. Açò però, voldria dir que $p \mid (i-j)a$ i com que a no és divisible per p tindrem que $p \mid i - j$. Pel fet que i i j són menors que p, l'única opció és que $i = j$. Per tant, $a, 2a, \ldots, (p-1)a$ són senzillament una reordenació de $1, 2, \ldots, p-1$ quan es considera el mòdul p. Com a conseqüència, el producte dels nombres en la 1a seqüència són congruents mòdul p al producte dels nombres de la 2a seqüència. És a dir, $a^{p-1}(p-1)! \equiv (p-1)! \bmod p$.

□

Aquest teorema fou enunciat per primera vegada l'any 1640 per Pierre Fermat a una carta dirigida a Frénicle de Bessy [8].

Corol·lari 4. *Si* p *no divideix* a *i si* $n \equiv m \bmod (p-1)$, *aleshores* $a^n \equiv a^m \bmod p$.

Demostració. Siga $n > m$. Com que $p - 1 \mid n - m$, tenim que $n = m + c(p-1)$ per a algun enter positiu c. Si multipliquem la congruència $a^{p-1} \equiv 1 \bmod m$ per si mateixa c vegades i després per $a^m \equiv a^m \bmod p$ obtenim el resultat desitjat. □

Un exemple d'ús de l'anterior resultat és el següent.

Exemple 4. Trobar l'últim dígit en base 7 de $2^{1000000}$.

En aquest cas, $p = 7$, com que $\frac{1000000}{7-1} = q_1 \cdot 6 + 4$ aleshores, $1000000 \equiv 4 \bmod 6$. Per tant, com que 7 no divideix a 2,

$$\begin{aligned} 2^{1000000} &\equiv 2^4 \bmod 7 \\ 2^4 &\equiv 2 \bmod 7. \end{aligned}$$

Així doncs, l'últim dígit en base 7 de $2^{1000000}$ és 2.

A l'hora de treballar amb més d'una equació de congruència, el teorema del residu xinés és un dels resultats més útils. Aquest teorema fou enunciat per primera vegada al segle III [26] a un escrit xinés, però no fou fins l'any 1247 quan es va enunciar de manera generalitzada junt amb la seva demostració [18]. No hi va haver una traducció a l'anglès fins el segle XIX.

Proposició 6 (Teorema del residu xinés)**.** *Es considera un sistema de congruències amb diferents mòduls*

$$\begin{aligned} x \equiv & a_1 \bmod m_1 \\ x \equiv & a_2 \bmod m_2 \\ & \ldots \\ x \equiv & a_r \bmod m_r \end{aligned}$$

amb cada parella primers entre sí, i.e. m.c.d.$(m_i, m_j) = 1$ *per a* $i \neq j$*. Aleshores existeix una solució simultània a totes les congruències i qualssevol* 2 *solucions són congruents a un altre mòdul:* $M = m_1 m_2 ... m_r$*.*

Demostració. En primer lloc, demostrarem la unicitat mòdul M.

Suposem que x', x'' són dues solucions al sistema de congruències. Siga $x = x' - x''$, aleshores $x \equiv 0 \bmod m_i$, $i \in \{1, 2, 3, ..., r\}$. Com a conseqüència $x \equiv 0 \bmod M$. A continuació mirarem de construir una solució.

Siga $M_i = M/m_i$, és a dir, el producte de tots els mòduls m_i excepte del i-èssim. Tenim que m.c.d$(m_i, M_i) = 1$, i per tant existeix un enter N_i (que es pot trobar mitjançant l'algoritme Euclidià) tal que $M_i N_i \equiv 1 \bmod m_i$.

Siga $x = \sum_i a_i M_i N_i$, per a cada i tenim que els termes en la suma diferents del terme i-èssim són tots divisibles per m_i perquè $m_i \mid M_j$ sempre que $j \neq i$. Així, per a cada i tenim que els termes en la suma diferents del terme i-èssim són tots divisible per m_i. Per tant, per a cada i tenim: $x \equiv a_i M_i N_i \equiv a_i \bmod m_i$. □

Recordant la definició de la funció d'Euler, tenim el següent corol·lari que ens servirà per a demostrar una proposició que s'usarà més endavant quan es discuteixi el sistema de clau pública de criptografia RSA.

Corol·lari 5. *La funció Euler és multiplicativa, en el sentit que* $\varphi(mn) = \varphi(m)\varphi(n)$ *si m.c.d.*$((m, n) = 1$*.*

Com que tot natural n es pot escriure com a producte de potències de nombres

primers, cadascuna de les quals no té factors comuns amb les altres i com que coneixem la fórmula $\varphi(p^\alpha) = p^\alpha\left(1-\frac{1}{p}\right)$, es pot usar el corol·lari anterior per a concloure que per a $n = p_1^{\alpha_1}p_2^{\alpha_2}\cdots p_r^{\alpha_r}$:

$$\varphi(n) = p_1^{\alpha_1}\left(1-\frac{1}{p_1}\right)p_2^{\alpha_2}\left(1-\frac{1}{p_2}\right)\cdots p_r^{\alpha_r}\left(1-\frac{1}{p_r}\right) = n\prod_{p|n}\left(1-\frac{1}{p}\right).$$

Com a conseqüència de la fórmula per a $\varphi(n)$, tenim el següent fet.

Proposició 7. *Siga n el resultat del producte de dos nombres primers diferents. Aleshores, saber quins són els dos primers p, q és equivalent a saber $\varphi(n)$. De manera més precisa, es pot calcular $\varphi(n)$ amb p, q en $O(\log n)$ operacions bit, i es pot calcular p i q amb n i $\varphi(n)$ en $O\left(\log^3 n\right)$ operacions bit.*

Demostració. Si n és parell, la proposició és trivial perquè llavors sabem que $p = 2$, $q = n/2$ i $\varphi(n) = n/2 - 1$.

Suposem doncs que n és imparell. Com que φ és multiplicativa, per a $n = pq$ tenim $\varphi(n) = (p-1)(q-1) = n + 1 - (p+q)$. Així, podem obtindre $\varphi(n)$ amb només restes i una suma i per tant aquest càlcul seria $O(\log n)$. D'altra banda, si coneixem n i $\varphi(n)$, sabem que:

$$\begin{aligned} p + q &= n + 1 + \varphi(n), \\ pq &= n. \end{aligned}$$

Llavors com que n és imparell, també ho han de ser p i q, de manera que $p + q = 2b$ és parell. Però dos nombres que compleixen les anteriors condicions han de ser les arrels de l'equació quadràtica $x^2 - 2bx + n = 0$. Per tant, p i q són igual a $b \pm \sqrt{b^2 - n}$. La part que més temps requereix és el càlcul de l'arrel quadrada que es pot fer amb $O\left(\log^3 n\right)$ operacions bit. □

Proposició 8 (Generalització del petit teorema de Fermat)**.** *Si m.c.d.$(a, m) = 1$, aleshores $a^{\varphi(m)} \equiv 1 \bmod m$.*

Demostració. En primer lloc, es veurà la demostració per al cas que m és una potència primera ($m = p^\alpha$). Aquest cas es pot demostrar per inducció.

- Per a $\alpha = 1$ es compleix per la segona part del petit teorema de Fermat perquè en aquest cas $\varphi(p) = p - 1$.
- Suposem que es compleix per a $\alpha - 1$.

- Per hipòtesi d'inducció $a^{p^{\alpha-1}-p^{\alpha-2}} = 1 + p^{\alpha-1}b$ per a algun enter b. Si elevem ambdues parts de l'equació a p, llavors tenim que

  $$a^{p^\alpha - p^{\alpha-1}} = 1 + (p^{\alpha-1}b)^p + \sum_{k=1}^{p-1} \binom{p}{k} 1^k (p^{\alpha-1}b)^{p-k}.$$

  Tots els termes de la suma resultant són divisibles entre p^α excepte el primer. Així doncs, $a^{p^\alpha - p^{\alpha-1}} - 1$ és divisible entre p^α, i per tant queda demostrat per a α perquè $\varphi(p^\alpha) = p^\alpha - p^{\alpha-1}$.

Així doncs, en cas que m siga un producte de potències de factors primers diferents qualsevol, per a cada potència primera p^α tal que $p^\alpha \mid\mid m$ es complirà que $a^{\varphi(p^\alpha)} \equiv 1 \mod p^\alpha$. Tenint en compte que φ és multiplicativa, $a^{\varphi(m)} \equiv 1 \mod p^\alpha$. I, per la propietat 5 de la Proposició 3, $a^{\varphi(m)} \equiv 1 \mod p^\alpha$. Amb açò queda demostrada la proposició. □

Aquesta proposició se li atribueix a Euler. De la demostració d'aquesta proposició podem obtindre un mètode per a obtindre una potència més menuda de a que garanteix ser congruent amb 1. Aquesta potència és el mínim comú múltiple de les potències p^α que donen $1 \mod p^\alpha$. Veiem com utilitzar açò amb el següent exemple.

Exemple 5. Calcular $2^{1000000} \mod 77$.

$77 = 7 \cdot 11$, la potència que dona $1 \mod 7$ és $7 - 1 = 6$ i per a 11 és 10. El m.c.m.$(6, 10) = 30$. Llavors $2^{30} \equiv 1 \mod 77$.
D'altra banda, com que $1000000 = 30 \cdot 33333 + 10$ aleshores tenim que $2^{1000000} \equiv 2^{10} \equiv 23 \mod 77$.

El següent procediment permet, d'una manera molt eficient, elevar valors a una potència en un anell de congruències. Aquest procediment serà essencial en algoritmes posteriors.

4.1 Exponenciació modular mitjançant el mètode de l'elevació al quadrat de manera repetida

Un problema bàsic en l'aritmètica modular és el de trobar $b^n \mod m$ quan m i n són grans. Una alternativa més eficient a calcular b^n i obtindre el seu residu -procediment

impossible de realitzar amb valors grans- és l'algoritme que s'exposa a continuació. Assumint que $b < m$, l'algoritme es bassa en cada passa que es fa una multiplicació es redueix mod m de manera que mai trobarem amb un enter major que m^2.

1. Siga $a = 1$, $b_0 = b$, i siguen $n_0, n_1, ..., n_{k-1}$ els dígits binaris de n.
2. Cada n_j és o 0 o 1. Si $n_j = 0$, es manté a igual, si $n_j = 1$ $a = a \cdot b_j$.
3. Es redueix $a \mod m$, i $b_{j+1} = b_j^2 \mod m$.
4. Es repeteixen els passos 2 i 3 fins a arribar a l'etapa $k - 1$.

Al finalitzar aquest algoritme, $a \equiv b^n \mod m$. Amb aquest algoritme es realitzen $k - 1$ etapes, cadascuna requereix una o dues multiplicacions de nombres $< m^2$. És a dir, cada etapa requereix $O(\log^2 m)$. Com que es fan tantes etapes com xifres binàries té n, el cost total d'aquest algoritme és de $O(\log n \cdot \log^2 m)$.

Referències. En aquest capítol s'han usat principalment l'obra de G. A. Jones i J. M. Jones *Elementary number theory* [16], H. Cohen *A course in computational algebraic number theory* [10] i l'obra de N. Koblitz [20].

Capítol 5

Factorització de nombres enters

En aquesta secció es veuran alguns resultats útils per a la factorització de nombres enters.

Proposició 9. *Siga b un nombre primer amb m i tal que $b^a \equiv 1 \mod m$, i $b^c \equiv 1 \mod m$ per a $a, c \in \mathbb{N}^*$. Aleshores $b^d \equiv 1 \mod m$, on $d =$m.c.d.(a, c).*

Demostració. Anteriorment ja s'ha vist que l'algoritme d'Euclides dona lloc a la igualtat $d = ua + vc$ amb $u, v \in \mathbb{Z}$. Com que d és el màxim comú divisor de a i c, $u > 0$ i $v \leq 0$ o a l'inrevés. Així doncs, sense pèrdua de la generalitat assumim que $u > 0$ i $v \leq 0$.

Si elevem ambdues parts de la congruència $b^a \equiv 1 \mod m$ a u, i fem el mateix per a la congruència $b^c \equiv 1 \mod m$ elevant-la a $-v$, obtenim

$$b^{au} \equiv 1 \mod m,$$
$$b^{-cv} \equiv 1 \mod m.$$

Si dividim ambdós congruències obtenim el resultat desitjat: $b^{au+cv} \equiv 1 \mod m$. □

Proposició 10. *Siga p un nombre primer que divideix a $b^n - 1$. Aleshores o i) p divideix a $b^d - 1$ per a algun divisor propi d de n, o ii) $p \equiv 1 \mod n$. En aquest últim cas, si a més a més $p > 2$ i n és imparell, aleshores $p \equiv 1 \mod 2n$.*

Demostració. D'una banda, tenim que $b^n \equiv 1 \mod p$. D'altra banda, pel petit teorema de Fermat, tenim que $b^{p-1} \equiv 1 \mod p$ (cal tenir en compte que b no és divisible entre p perquè $p \mid b^n - 1$). Per l'anterior proposició, això implica que $b^d \equiv 1 \mod p$ on $d =$m.c.d.$(n, p - 1)$.

Si $d < n$ es compleix i). Si $d = n$, $p \equiv 1 \mod n$. En aquest últim cas, si $p > 2$, p i n són imparells. Per tant, $p-1$ és parell i com que és divisible entre n, $p \equiv 1 \mod 2n$. $\square$

L'anterior proposició pot servir d'ajuda per a factoritzar un nombre tal i com veurem amb els següents exemples.

Exemple 6. Factoritzar $3^{12} - 1 = 531440$.

Per la proposició anterior, sabem que si p no divideix a $3^d - 1$ (on d és un divisor propi de 12 en aquest cas), aleshores p tampoc dividirà a $3^{12} - 1$. Així doncs, provem amb els divisors menors de 12, i.e. $d \in \{1, 2, 3, 4, 6\}$. Tenim

$$3 - 1 = 2$$
$$3^2 - 1 = 2^3$$
$$3^3 - 1 = 2 \cdot 13$$
$$3^4 - 1 = 2^4 \cdot 5$$
$$3^6 - 1 = 2^2 \cdot 7.$$

Els factors obtinguts $2^4, 5, 7, 13$ divideixen en aquest cas a $3^{12} - 1$. Finalment, fent la divisió $\frac{3^{12}-1}{2^4 \cdot 5 \cdot 7 \cdot 13} = 73$, que és un nombre primer corresponent a l'últim factor que ens quedava per trobar. Tal i com afirma la proposició, $73 \equiv 1 \mod 12$.

Exemple 7. Trobar la factorització de $2^{35} - 1$.

Procedirem de la mateixa manera que en l'anterior exemple buscant els de la forma $2^d - 1$ on $d \in \{1, 5, 7\}$. En aquest cas obtenim 31 i 127. La proposició ens assegura que els següents divisors seran $\equiv 1 \mod 35\cdot$. Així només cal buscar els divisors de $\frac{2^{35}-1}{31 \cdot 127} = 8727391$ que siguen $\equiv 1 \mod 70$. Per tant, s'ha de provar amb $71, 211, 281, \ldots$. A priori s'haurà de comprovar que tots els primer menors que $\sqrt{8727391} = 2954$. Resulta però, que $8727391 = 71 \cdot 122921$. Així, només cal comprovar com a màxim $\sqrt{122921} = 350$ nombres per a descobrir que 122921 és un nombre primer i per tant, la factorització és $2^{35} - 1 = 31 \cdot 127 \cdot 71 \cdot 122921$.

Referències. En aquest capítol s'han aprofitat l'obra de N. Koblitz [20], G. A. Jones i J. M. Jones [16] i G. H. Hardy i E. M. Wright [14].

Capítol 6

Sistemes d'encriptació

L'objectiu de la criptografia és el de proporcionar un sistema per tal que dues entitats es puguen enviar missatges de manera confidencial a través d'un canal no segur amb la finalitat que un oponent (en cas d'accedir al text enviat) no puga entendre el missatge. A continuació es presenta de manera més formal la definició d'un sistema d'encriptació.

Definició 9. *Un sistema d'encriptació és una cinc-tupla* $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$*, que compleix les següents condicions:*

1. $\mathcal{P}$ *és un conjunt finit de possibles texts.*

2. $\mathcal{C}$ *és un conjunt finit de possibles texts xifrats.*

3. $\mathcal{K}$ *és un conjunt finit amb les possibles claus.*

4. *Per a cada clau* $K \in \mathcal{K}$*, existeix una regla de xifratge* $e_K \in \mathcal{E}$ *amb la seva corresponent regla de desxifratge* $d_K \in \mathcal{D}$*. Cada* $e_K : \mathcal{P} \to \mathcal{C}$ *i* $d_K : \mathcal{C} \to \mathcal{P}$ *són funcions tal que* $d_K(e_K(x)) = x$ *per a tot element del text original* $x \in \mathcal{P}$*.*

Cada funció d'encriptació e_K ha de ser injectiva, en cas contrari, no es podria desxifrar el missatge de manera clara.

6.1 Sistemes d'encriptació de clau privada

Els sistemes d'encriptació de clau privada o simètrics són aquells que usen la mateixa clau per a encriptar i desencriptar el missatge. Així, l'emissor i el receptor comparteixen la mateixa clau per a encriptar i desencriptar missatges i enviar-se'ls entre si. A més a més, és essencial que aquesta clau siga secreta per tal de poder mantenir una comunicació confidencial. Per tant, les dues entitats implicades en la comunicació han de trobar la manera segura de transmetre's la clau. Aquest tipus de sistemes d'encriptació foren els primers a usar-se.

A continuació mostrem l'exemple més clàssic i senzill de xifratge de clau privada.

Exemple 8. Un dels primers sistemes d'encriptació usat fou el de xifratge amb decalatge. En aquest, $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_m$. Per a cada llengua, m denota la quantitat de caràcters diferents (ignorant en aquest cas per simplicitat els espais en blanc). Per a l'anglès $m = 26$, mentre que per al català $m = 35$ i el castellà $m = 32$. Així, per al cas de l'anglès, es defineix $0 \leq K \leq 25$ junt amb les funcions de d'encriptació

$$e_K(x) = (x + K) \bmod 26,$$

i desencriptació

$$d_K(y) = (y - K) \bmod 26.$$

$$(x, y \in \mathbb{Z}_{26})$$

Per a poder fer aquest tipus d'operacions, a cada caràcter de la llengua se li assigna un nombre per ordre alfabètic començant en 0. Així, per exemple per a $K = 11$, després de passar-li la funció d'encriptació el missatge "wewillmeetatmidnight"passaria a "hphtwwxppelextoytrse".

Aquest sistema, malgrat haver sigut usat per personatges històrics com Juli Cèsar (qui utilitzava la clau 3) [17], no té gaire sentit en l'actualitat ja que és bastant ràpid trobar la clau K usada només intentant desxifrar el mateix missatge amb totes les claus possibles. A més a més, en un sistema d'encriptació on hi hagueren més possibles claus i per tant no fora factible provar-les totes, però (de la mateixa manera que en l'exemple) un mateix caràcter es xifrara sempre amb un altre mateix caràcter (diferent a l'original) es podria desxifrar de manera senzilla usant el coneixement de la llengua emprada. Per exemple, el caràcter més usat en un text escrit en anglés és la lletra e, seguida per lletres com la t, s o r entre d'altres. Això, junt amb les parelles i trios de caràcters més abundants en una llengua permeten desxifrar un missatge sense necessitat de conéixer la clau.

Un exemple però, de xifratge que no sempre assigna el mateix caràcter per a una lletra determinada és el xifratge de Vigenère.

Definició 10. *Siga m un enter positiu. Siguen $\mathcal{P} = \mathcal{C} = \mathcal{K} = (\mathbb{Z}_{26})^m$. Per a una clau $K = (k_1, k_2, \ldots, k_m)$, es defineix*

$$e_K(x_1, x_2, \ldots, x_m) = (x_1 + k_1, x_2 + k_2, \ldots, x_m + k_m)$$

i

$$\boldsymbol{d}_K(y_1, y_2, \ldots, y_m) = (y_1 - k_1, y_2 - k_2, \ldots, y_m - k_m),$$

Totes les operacions es realitzen en $\mathbb{Z}_{26}$.

Exemple 9. Si $m = 6$, amb clau $K = (2, 8, 15, 7, 4, 17)$ i el text a encriptar és "thiscryptosystemisnotsecure", cal convertir primer tots els caràcters a la seva representació numèrica en $\mathbb{Z}_{26}$ i seguidament, agrupar-los de sis en sis per tal de sumar-los la clau.

19	7	8	18	2	17	24	15	19	14	18	24
2	8	15	7	4	17	2	8	15	7	4	17
21	15	23	25	6	8	0	23	8	21	22	15

18	19	4	12	8	18	13	14	19	18	4	2
2	8	15	7	4	17	2	8	15	7	4	17
20	1	19	19	12	9	15	22	8	25	8	19

20	17	4
2	8	15
22	25	19

Finalment, cal reconvertir els valors numèrics obtinguts a la seua representació gràfica obtenint el missatge xifrat "vpxzgiaxivwpubttmjpwizitwzt".

Per al cas de Vigenère, desxifrar el missatge provant les 26^m claus possibles no seria gaire viable a mà. Tot i així, encara es podria realitzar mitjançant un ordinador.

6.1.1 DES i AES

L'interès per encriptar missatges de manera efectiva (i. e., complint el seu propòsit de no poder ésser desencriptats), ha sigut motiu d'estudi dels diferents atacs que es poden realitzar a un sistema d'encriptació i les maneres d'evitar-los. L'any 1977 es va

publicar el primer estàndard de sistema d'encriptació per la *National Bureau of Standards*, actualment *National Institute of Standards and Technology* (NIST): el *Data Encryption Standard* (conegut per les seves inicials com a DES). Aquest sistema era segur front als atacs coneguts fins aleshores per la IBM. Consistia essencialment en permutacions i substitucions, el seu espai de claus consistia de 2^{56} claus possibles. Cada clau tenia 8 bits de paritat, de manera que cada clau era de 64 bits en total. Amb el temps es va arribar a usar molt i la NIST el revisava cada 5 anys. Entre els atacs que es van estudiar per a aquest estàndard, es troba el de la força bruta. En el moment del seu naixement, per tal d'atacar aquest sistema amb aquest mètode es requeria una màquina de 20.000.000 dòlars. Malgrat que aquesta màquina podia desencriptar el missatge en un dia, el seu cost feia l'atac poc factible. No obstant això, amb els anys la tecnologia va evolucionar i l'any 1993 Michael Wiener va publicar una manera detallada de construir una màquina capaç de desencriptar un missatge amb el mètode de la força bruta en un dia i mig i amb un cost de 100.000 dòlars [25]. Amb açò, es va passar a buscar una alternativa al DES. Així, l'any 1999 la NIST va revisar per última vegada el DES i es va publicar el 3DES o *Triple DES*. Aquest algoritme aplicava 3 vegades l'algoritme del DES. El problema d'aquest nou sistema però, és que era lent.

Simultàniament, l'any 1997 la NIST va anunciar un concurs públic per tal de trobar un substitut al DES. Les condicions d'aquest sistema era que la longitud del bloc fora de 128 bits, i la longitud de les claus de 128, 192 i 256 bits. A més a més, el sistema havia d'estar públic per a tothom. Els guanyadors d'aquest concurs per la seguretat, el cost i les característiques d'implementació de l'algoritme foren Daemen i Rijmen. Aquest sistema es va adoptar com a estàndard per la NIST l'any 2001 amb el nom de *Advanced Encryption Standard* o AES.

L'AES és un sistema de blocs de 128 bits, i.e. per a encriptar un missatge es divideix en grups de 128 bits i s'aplica l'algoritme. Segons si el tamany de la clau és de 128, 192 o 256 bits, l'algoritme consta de 10, 12 o 14 rondes respectivament. Així, si la quantitat de rondes és n, en la part d'inicialització es generen $n+1$ claus de 128 bits per a cada ronda (més la d'inicialització) a partir de la clau original. A més a més, es representa el bloc original en una matriu de 4×4 bytes anomenada matriu d'estats. Llavors, s'aplica un *or* exclusiu de la matriu d'estats amb la clau k_0.

Seguidament per a cada ronda, s'apliquen 4 transformacions a la matriu d'estats de manera consecutiva: substitució, desplaçament de files, barreja de columnes i *or* exclusiu amb la següent clau. L'operació de substitució ja l'hem vista i consisteix en reemplaçar uns bytes per uns altres. En segon lloc, el desplaçament de files deixa la primera fila fixa, la segona la mou cap a l'esquerra una posició de manera que el primer element de la fila passa a ser l'últim i amb la mateixa mecànica la tercera fila es desplaça dues

posicions i la quarta tres. En tercer lloc, la barreja de columnes reemplaça cada columna de la matriu per una altra resultat de multiplicar una matriu predeterminada amb la columna. Finalment, després de seguir aquest procediment per a les n rondes, obtenim el text encriptat.

Actualment, l'AES s'usa en nombroses situacions des de transaccions financeres fins a protecció de dades confidencials per part del govern o l'exèrcit. Tanmateix, una de les aplicacions més quotidianes de l'AES es troba en les xarxes. Així, quan hom es connecta a una xarxa wifi privada, li cal la clau (o contrasenya) per a poder accedir-hi i rebre o enviar dades a través de la xarxa.

6.2 Sistemes d'encriptació de clau pública

Com ja s'ha comentat, en els sistemes d'encriptació de clau privada, cal que les persones implicades en la comunicació acorden prèviament la clau K a usar abans de poder tindre una comunicació confidencial a través d'un canal insegur. Però açò pot no ser possible si les persones implicades es troben en llocs diferents. Per tal d'evitar aquests aspectes, es poden usar els sistemes d'encriptació de clau pública o asimètrics. Es diuen asimètrics perquè a diferència dels simètrics, hi ha dues claus: una per al procés d'encriptació i l'altra per al procés de desencriptació. La del procés d'encriptació s'anomena clau pública perquè està accessible a tothom i la del procés de desencriptació s'anomena clau privada. L'única entitat que té accés a la clau privada és el receptor dels missatges per tal de poder desencriptar-los. Així, tot el procés d'encriptació és conegut per tothom; fins i tot la clau requerida per aquest procés.

Per tal que la funció d'encriptació puga ser visible sense implicar problemes de seguretat, ha de complir certes condicions. En primer lloc, cal que aquesta funció sigui fàcil d'aplicar però difícil d'invertir. En segon lloc, per tal d'invertir aquesta funció cal que hi haja un camí alternatiu secret que permeta la desencriptació del missatge. Aquest tipus de funcions es coneixen en l'anglès amb el nom de *trapdoor one-way functions*.

L'exemple per excel·lència de sistema d'encriptació de clau pública és el RSA (Rivest-Shamir-Adleman). Aquest sistema es basa en la dificultat de trobar la factorització d'un nombre enter suficientment gran. Tot i així, hi ha altres sistemes d'encriptació de clau pública com els logarítmics. Aquests es basen en la dificultat de calcular el logaritme discret (funció logaritme però en grups) en alguns grups en particular com són els subgrups d'un ordre primer gran. Exemples de sistemes que es basen en aquest problema són

l'intercanvi de claus Diffie-Hellman [11] o el sistema d'encriptació ElGamal[2].

6.2.1 RSA

El sistema RSA és un dels més usats avui en dia, sobretot pel que fa al tràfic d'Internet, i ve definit de la següent manera:

Definició 11. *Siga $n = pq$, on p i q són primers. Siga $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n$, es defineix*

$$\mathcal{K} = \{(n, p, q, a, b) : ab \equiv 1(\mathrm{mod}\varphi(n))\}.$$

Per a $K = (n, p, q, a, b)$, es defineixen

$$e_K(x) = x^b \bmod n$$

i

$$d_K(y) = y^a \bmod n$$

$(x, y \in \mathbb{Z}_n)$. La parella formada per n i b és la clau pública, mentre que p, q i a formen la clau privada.

Per a comprovar que el sistema funciona, en primer lloc verificarem que les funcions d'encriptació i desencriptació són inverses.

D'una banda,

$$ab \equiv 1 \bmod \varphi(n)$$
$$ab = t\varphi(n) + 1 \text{ per a algun } t \in \mathbb{Z} \text{ amb } t \geq 1.$$

Si $x \in \mathbb{Z}_n^*$ (on $\mathbb{Z}_n^*$ denota el conjunt de $x \in \mathbb{Z}_n$ que són coprimers a n), aleshores

$$\begin{aligned}\left(x^b\right)^a &\equiv x^{t\varphi(n)+1} \bmod n \\ &\equiv \left(x^{\varphi(n)}\right)^t x \bmod n \\ &\equiv 1^t x \bmod n \text{ per la generalització del petit teorema de Fermat} \\ &\equiv x(\mathrm{mod}n).\end{aligned}$$

Si $x \in \mathbb{Z}_n$ no és coprimer a n, considerem $d = m.c.d.(x, n)$. Com que $n = pq$ amb p i q primers, o $d = p$ o $d = q$. Suposem que $d = p$, llavors

$$\begin{aligned}\left(x^b\right)^a &\equiv x^{t\varphi(n)+1} \bmod n \\ &\equiv p^{t\varphi(n)+1} k^{t\varphi(n)+1} \bmod pq \text{ per a algun } k \in \mathbb{Z}_n \\ &\equiv p^{t\varphi(n)} k^{t\varphi(n)+1} \bmod q.\end{aligned}$$

Com que p i q són primers i $m.c.d.(x,n) = p$, k és coprimer amb q.

$$\begin{aligned}\left(x^b\right)^a &\equiv p^{t\varphi(n)}k^{t\varphi(n)+1} \bmod q\\ &\equiv 1^{t\varphi(p)}1^{t\varphi(p)}k \bmod q\\ &\equiv k \bmod q\\ &\equiv pk \bmod pn\\ &\equiv x \bmod n.\end{aligned}$$

En cas que $d = q$ el procediment seria el mateix. Amb açò, hem demostrat que aplicant la funció de desencriptació al missatge xifrat, hom obtindrà el missatge original.

Veiem un exemple concret d'encriptació i desencriptació amb el RSA. Un exemple pràctic seria anàleg amb valors de p i q de més de 1024 bits.

Exemple 10. Siga $p = 101$ i $q = 113$, aleshores $n = 11413$ i $\varphi(n) = (p-1)(q-1) = 11200$. Si triem $b = 3533$, el primer pas és comprovar que $m.c.d.(b, \varphi(n)) = 1$. Per a això es pot usar l'algoritme d'Euclides tal i com s'ha vist anteriorment.

$$\begin{aligned}11200 &= 3 \cdot 3533 + 601\\ 3533 &= 5 \cdot 601 + 528\\ 601 &= 1 \cdot 528 + 73\\ 528 &= 7 \cdot 73 + 17\\ 73 &= 4 \cdot 17 + 5\\ 17 &= 3 \cdot 5 + 2\\ 5 &= 2 \cdot 2 + 1.\end{aligned}$$

Per tal de desxifrar els missatges que li arriben a l'emissor, cal determinar $a = b^{-1} \bmod 11200$. Açò es pot fer aprofitant el procediment anterior:

$$\begin{aligned}1 &= 5 - 2 \cdot 2\\ &= 7 \cdot 5 - 2 \cdot 17\\ &= 7 \cdot 73 - 30 \cdot 17\\ &= 217 \cdot 73 - 30 \cdot 528\\ &= 217 \cdot 601 - 247 \cdot 528\\ &= 1452 \cdot 601 - 247 \cdot 3533\\ &= 1452 \cdot 11200 - 4603 \cdot 3533\\ &= 1452 \cdot 11200 + 6597 \cdot 3533.\end{aligned}$$

Per tant, $a = 6597$. I, per a xifrar el missatge 9726, es calcula $9726^{3533} \bmod 11413 = 5761$

mitjançant l'algoritme d'elevació al quadrat de manera repetida (vist anteriorment). Açò es descriu en la següent graella.

i	b_i	z
0	1	$1^2 \cdot 9726 = 9726$
1	1	$9726^2 \cdot 9726 = 2659$
2	0	$2659^2 = 5634$
3	1	$5634^2 \cdot 9726 = 9167$
4	1	$9167^2 \cdot 9726 = 4958$
5	1	$4958^2 \cdot 9726 = 7783$
6	0	$7783^2 = 6298$
7	0	$6298^2 = 4629$
8	1	$4629^2 \cdot 9726 = 10185$
9	1	$10185^2 \cdot 9726 = 105$
10	0	$105^2 = 11025$
11	1	$11025^2 \cdot 9726 = 5761$

Per a desxifrar, es realitza mitjançant el mateix algoritme el càlcul de $5761^{6597} \bmod 11413 = 9726$, obtenint el resultat esperat.

6.2.2 Cost computacional del sistema d'encriptació RSA

Totes les operacions que cal dur a terme per tal de poder posar a la pràctica el sistema RSA es poden resumir en el següent llistat:

- Generar p i q primers.
- Calcular $n = pq$ i $\varphi(n) = (p-1)(q-1)$.
- Generar $1 < b < \varphi(n)$ tal que $m.c.d.(b, \varphi(n)) = 1$.
- Calcular $a = b^{-1} \bmod \varphi(n)$.
- Calcular $x^b \bmod n$ si s'està encriptant o $y^a \bmod n$ per a desencriptar.

Ja hem vist que l'algoritme per a realitzar les operacions d'encriptació i desencriptació són d'ordre $O(\log b \log^2 n)$ i $O(\log a \log^2 n)$ respectivament. A més a més, calcular a costa

$O(\log^3 b)$ i $n = pq$ i $\varphi(n) = (p-1)(q-1)$ són d'ordre $O(\log^2 n)$. Amb això, només queda la generació dels paràmetres p, q i b que analitzarem amb més deteniment a continuació.

En primer lloc, estudiarem la generació dels nombres primers p i q. A l'hora de generar-los, cal tenir en compte que si un oponent pot arribar a factoritzar n el sistema ja no és segur. Per tant, p i q han de ser suficientment grans com per a poder impedir aquesta factorització. Com que els algoritmes de l'actualitat poden factoritzar amb facilitat nombres de fins a 512 bits, es solen agafar p i q primers la representació binària dels quals siga aproximadament de 512 bits. Així, n té 1024 bits aproximadament.

Per a generar p i q, es genera un nombre aleatori i es comprova que aquest és primer de manera reiterada fins obtindre dos nombres primers. Així, per tal de saber el cost de la generació d'aquests dos nombres, primer cal determinar la quantitat de nombres aleatoris que generarem abans d'arribar a un primer. Si denotem per $\pi(n)$ a la quantitat de nombres primers menors o iguals que n, el teorema dels nombres primers afirma que $\pi(n)$ és aproximadament $\frac{n}{\log n}$. Així, la probabilitat que un nombre enter $1 < p < n$ de 512 bits siga primer és $\frac{1}{\log 2^{512}}$. Si a més a més, considerem només els nombres senars, la probabilitat es duplica a $\frac{2}{\log 2^{512}}$.

Pel que fa a la comprovació de que el nombre en qüestió siga primer, malgrat que l'any 2002 es va demostrar per Agrawal, Kayal i Saxena que existia un algoritme amb temps polinòmic determinista per a testejar si un nombre és primer, en l'actualitat el procediment a seguir sol usar encara algoritmes polinòmics aleatoris de tipus Monte Carlo. Aquests algoritmes tenen l'avantatge de ser ràpids ($O(\log n)$), hi ha la possibilitat però, de no obtindre la resposta correcta mitjançant aquest algoritmes. És per això que es sol córrer l'algorisme més d'una vegada fins que la probabilitat d'error es troba per sota d'un llindar definit. Un exemple d'aquest tipus d'algoritmes és l'algoritme Miller-Rabin:

1. Siga $n - 1 = 2^k m$, on m és imparell.
2. Tria un enter $a, 1 \leq a \leq n-1$ de manera aleatòria.
3. Siga $b = a^m \bmod n$.
4. Si $b \equiv 1 \bmod n$ aleshores retorna "n és primer".
5. En cas contrari, itera des de $i = 0$ fins a $i = k - 1$:
   - Si $b \equiv -1 \bmod n$, aleshores retorna "n és primer".
   - Si no, $b = b^2 \bmod n$.
6. Si no s'ha retornat cap resultat, retorna "n és compost".

L'algoritme Miller-Rabin és un algoritme de decisió que intenta respondre a la pregunta de si el nombre generat n és compost o no. És de tipus Monte Carlo sí-esbiaixat. És a dir, sempre que s'afirme que n és compost serà veritat, però si n és compost l'algoritme afirmarà que és primer amb probabilitat $\frac{1}{4}$. Seguidament, demostrarem que l'algoritme és sí-esbiaixat.

Teorema 2. *L'algoritme Miller-Rabin per a composats és un algoritme Monte Carlo sí-esbiaixat.*

Demostració. Si l'algoritme és sí esbiaixat, sempre que responga que n és compost ho serà. Suposem per contra que n és primer i l'algoritme retorna que n és compost. Per tal que retorne això, cal que $a^m \not\equiv 1 \bmod n$ i que per a tots els valors de b $(a^m, a^{2m}, \ldots, a^{2^{k-1}m})$ es compleixi que:

$$a^{2^i m} \not\equiv -1 \bmod n$$

per a $0 \leq i \leq k-1$.

D'altra banda, pel petit teorema de Fermat, $a^{2^k m} \equiv 1 \bmod n$. Llavors, $a^{2^{k-1}m}$ és una arrel quadrada de 1 mòdul n. Per ser n primer, les úniques arrels quadrades de 1 mòdul n són $\pm 1 \bmod n$. Com que

$$a^{2^{k-1}m} \not\equiv -1 \bmod n,$$

ha de ser

$$a^{2^{k-1}m} \equiv 1 \bmod n.$$

Però aleshores $a^{2^{k-2}m}$ ha de ser un altre cop arrel quadrada de 1 mòdul n. Pel mateix argument,

$$a^{2^{k-2}m} \equiv 1 \bmod n.$$

Reiterant aquest argument, arribem a

$$a^m \equiv 1 \bmod n.$$

Però açò és una contradicció perquè en aquest cas haguera contestat que n és primer. □

Així doncs, s'hauran de generar $\log 2^{512} \approx 355$ nombres per tal de trobar un nombre primer. Per cada nombre generat, l'algoritme que testeja si és compost o no és d'ordre polinòmic.

6.2.3 Criptoanàlisi del mètode RSA

A l'hora d'atacar un sistema criptogràfic RSA, un primer intent és el de factoritzar n. Així, si obtenim els factors primers p i q, podem obtindre de manera immediata $\varphi(n)$ i

amb això podem calcular a per tal de poder desxifrar qualsevol missatge.

Els algoritmes més usats en l'actualitat per a factoritzar un nombre són el garbell quadràtic, la factorització de corba el·líptica i el garbell sobre el cos dels nombres (*quadratic sieve, elliptic curve factoring* i *number field sieve* respectivament en anglés). Abans d'aquests però, era més habitual usar l'algoritme Pollard $p-1$ entre d'altres.

L'algoritme Pollard $p-1$ rep com a entrada el nombre n a factoritzar i una fita B i consta de les següents passes:

1. Inicialitzar $a = 2$, $j = 2$.
2. $a = a^j \bmod n$.
3. $j = j + 1$.
4. Repetir 2 i 3 fins arribar a $j = B$ inclosa.
5. $d = m.c.d.(a-1, n)$.
6. Si $1 < d < n$, retorna d, si no, retorna "error".

Aquest algoritme es bassa amb la següent idea. Siga p un nombre primer que divideix a n, suposem que $q \leq B$ per a tot q que divideix $p-1$. Aleshores $p-1 \mid B!$. Després del pas 4, tenim $a \equiv 2^{B!} \bmod n$. Com que $p \mid n$, $a \equiv 2^{B!} \bmod p$. Ara bé, pel petit teorema de Fermat, $2^{p-1} \equiv 1 \bmod p$.

D'altra banda, com que $p-1 \mid B!$, $a \equiv 1 \bmod p$. Llavors, $p \mid a-1$ i $p \mid n$. Per tant, $p \mid m.c.d.(a-1, n)$. En cas que n siga el producte de dos nombres primers (com és el cas en RSA), $p = d$.

Aquest algoritme realitza $B-1$ exponenciacions modulars, cadascuna amb $2 \log B$ multiplicacions si s'usa el mètode de l'elevació al quadrat. A més a més, com que es calcula al final el $m.c.d.$, l'ordre d'aquest algoritme és $O(B \log B \log^2 n + \log^3 n)$. El problema principal d'aquest algoritme és que trobar el factor de n depèn del valor de B. Si el valor de B és suficientment menut com per a fer el cost de l'algoritme polinòmic, la probabilitat de trobar un factor de n és molt baixa. Per contra, si augmentem el tamany de $B \approx \sqrt{n}$, no serà més ràpid que el procediment de trobar factors a força bruta. Així, per tal que l'algoritme siga eficient, n ha de tindre un factor primer p de manera que $p-1$ només tinga factors menuts. Per tant, hom pot esquivar l'atac amb aquest algoritme generant un nombre primer p_1 gran de manera que $p = 2p_1 + 1$ també siga primer (i el mateix amb $q = 2q_1 + 1$).

Altres mètodes més actuals com el garbell quadràtic usen el mètode de factorització de Fermat. Aquest, intenta trobar x, y tal que $x^2 - y^2 = n$ i així obtindre els dos factors $x + y$ i $x + y$.

Una primera aproximació és inicialitzar $x = [\sqrt{n}]$. Seguidament es comprova si $x^2 - n$ és un quadrat perfecte i s'incrementa x una unitat fins obtindre un quadrat perfecte. Aquest mètode però, pot requerir de moltes iteracions fins realment trobar un quadrat perfecte.

Per a acurtar aquest procediment Maurice Kraitchik va introduir un canvi. En aquest altre mètode es passa a l'aritmètica modular, de manera que es busquen x, y tal que $x^2 \equiv y^2 \bmod n$. Així, si $x^2 - y^2 = kn$ on $k \in \mathbb{Z}$, $m.c.d.(x - y, n)$ i $m.c.d.(x + y, n)$ són factors de n (sempre que n no divideixi ni $x+y$ ni $x-y$). Aquest mètode s'il·lustra millor amb un exemple.

Exemple 11. Si volem factoritzar 227179, procedirem igual que abans buscant un quadrat perfecte.

$477^2 - 227179 = 350 = 2 \cdot 5^2 \cdot 7$
$478^2 - 227179 = 1305 = 3^2 \cdot 5 \cdot 29$
$479^2 - 227179 = 2262 = 2 \cdot 3 \cdot 13 \cdot 29$
$480^2 - 227179 = 3221 = 3221$
$481^2 - 227179 = 4182 = 2 \cdot 3 \cdot 17 \cdot 41$
$482^2 - 227179 = 5145 = 3 \cdot 5 \cdot 7^3$
$483^2 - 227179 = 6110 = 2 \cdot 5 \cdot 13 \cdot 47$
$484^2 - 227179 = 7077 = 3 \cdot 7 \cdot 337$
$485^2 - 227179 = 8046 = 2 \cdot 3^3 \cdot 149$
$486^2 - 227179 = 9017 = 71 \cdot 127$
$487^2 - 227179 = 9990 = 2 \cdot 3^3 \cdot 5 \cdot 37$
$488^2 - 227179 = 10965 = 3 \cdot 5 \cdot 17 \cdot 43$
$489^2 - 227179 = 11942 = 2 \cdot 7 \cdot 853$
$490^2 - 227179 = 12921 = 3 \cdot 59 \cdot 73$
$491^2 - 227179 = 13902 = 2 \cdot 3 \cdot 7 \cdot 331$
$492^2 - 227179 = 14885 = 5 \cdot 13 \cdot 229$
$493^2 - 227179 = 15870 = 2 \cdot 3 \cdot 5 \cdot 23^2$
$494^2 - 227179 = 16857 = 3^2 \cdot 1873$

$495^2 - 227179 = 17846 = 2 \cdot 8923$
$496^2 - 227179 = 18837 = 3^2 \cdot 7 \cdot 13 \cdot 23.$

Després d'un nombre d'iteracions, busquem relacions entre els nombre obtinguts per tal de combinar-los i fer un quadrat perfecte. En aquest cas, els nombres que combinats ens donen un quadrat perfecte són el primer, el sisé i el desé.

$477^2 - 227179 = 350 = 2 \cdot 5^2 \cdot 7$
$482^2 - 227179 = 5145 = 3 \cdot 5 \cdot 7^3$
$493^2 - 227179 = 15870 = 2 \cdot 3 \cdot 5 \cdot 23^2.$
Si representem aquestes factoritzacions amb congruències obtenim el següent.

$477^2 \equiv 2 \cdot 5^2 \cdot 7 \bmod 227179$
$482^2 \equiv 3 \cdot 5 \cdot 7^3 \bmod 227179$
$493^2 \equiv 2 \cdot 3 \cdot 5 \cdot 23^2 \bmod 227179.$
Aquestes 3 factoritzacions combinades ens donen un quadrat perfecte.

$(477 \cdot 482 \cdot 493)^2 \equiv \left(2^2 \cdot 3^2 \cdot 5^4 \cdot 7^4 \cdot 23^2\right) \bmod 227179$
$(113347602)^2 \equiv \left(2 \cdot 3 \cdot 5^2 \cdot 7^2 \cdot 23\right)^2 \bmod 227179$
$(113347602)^2 \equiv (169050)^2 \bmod 227179.$

Com que 113347602, és major que 227179, cal reduir-lo mòdul 227179, obtenint $(212460)^2 \equiv (169050)^2 \bmod 227179$. Així, un factor és $m.c.d.(212460 - 169050, 227179) = 1447$ i l'altre es pot obtindre, o bé, calculant $m.c.d(212460 + 169050, 227179) = 157$, o bé, o dividint 227179 entre 1447.

El procés de buscar nombres que combinats ens donen un quadrat perfecte es pot descriure mitjançant l'àlgebra lineal per tal de poder realitzar-ho computacionalment. Així es forma una matriu amb els exponents de cadascun dels factors primers menors o iguals que un llindar B. Continuant amb l'exemple, si $B = 50$, la matriu tindrà 15 columnes corresponents als nombres primers que han aparegut en les descomposicions $\mathcal{B} = \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47\}$ i 20 files corresponents a cadascun dels nombres provats (veure 6.1).

Aquesta matriu es pot simplificar a l'hora de fer operacions amb ella si tenim en compte només la paritat. Així, les entrades parells seran 0 i les senars 1 (veure 6.2).

Ara, el que volem es trobar una relació lineal entre les files per tal d'obtindre una facto-

$$\begin{bmatrix}
1 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 3 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

Figura 6.1: Matriu inicial amb els exponents de cada primer per a cada nombre.

$$\begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

Figura 6.2: Matriu inicial amb la paritat dels exponents de cada primer per a cada nombre.

rització d'exponents parells, i.e. $[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]$. Dit d'una altra manera, el que estem buscant és una matriu S tal que $S \cdot M = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]$. Mitjançant el mètode de reducció de Gauss, podem obtindre la solució $S = [1,0,0,0,1,0,0,0,1,0]$, que es correspon amb el que ja havíem obtingut abans.

El conjunt $\mathcal{B}$ es sol anomenar base factor i de la matriu que forma es podrà obtindre una solució sempre que la quantitat de nombres primers siga menor que la quantitat de factoritzacions obtingudes.

En el nostre exemple, hem triat un nombre inicial x i després l'hem incrementat de manera reiterada una unitat. Una alternativa més eficient és considerar els enters de la forma $j + \sqrt{kn}$ per a $j, k \in \mathbb{N}$ o de la forma $\sqrt{kn}$.

Com ja s'ha dit, un dels algoritmes més usats és l'algoritme de garbell quadràtic.

Aquest usa alguns resultats que enunciarem a continuació.

Definició 12. *Siga p un nombre primer senar i a un enter. Es diu que a és un residu quadràtic mòdul p si $a \not\equiv 0 \bmod p$ i la congruència $y^2 \equiv a \bmod p$ té una solució $y \in \mathbb{Z}_p$. Per contra, es diu que a és un no-residu quadràtic mòdul p si $a \not\equiv 0 \bmod p$ i a no és un residu quadràtic mòdul p.*

Per tal de determinar si un nombre és un residu quadràtic mòdul p, s'usa el criteri d'Euler.

Teorema 3. *Siga p un nombre primer senar. Aleshores un enter a és un residu quadràtic mòdul p si i només si*

$$a^{(p-1)/2} \equiv 1 \bmod p.$$

Demostració. Suposem en primer lloc que a és un residu quadràtic mòdul p, és a dir, $y^2 \equiv a \bmod p$ té una solució $y \in \mathbb{Z}_p$.

Pel petit teorema de Fermat, $a^{p-1} \equiv 1 \bmod p$ per a qualsevol $a \not\equiv 0 \bmod p$. Aleshores,

$$\begin{aligned} a^{(p-1)/2} &\equiv \left(y^2\right)^{(p-1)/2} \bmod p \\ &\equiv y^{p-1} \bmod p \\ &\equiv 1 \bmod p. \end{aligned}$$

D'altra banda, si $a^{(p-1)/2} \equiv 1 \bmod p$. Siga b un element primitiu mòdul p. Aleshores $a \equiv b^i \bmod p$ per a algun enter positiu i. Per tant,

$$\begin{aligned} a^{(p-1)/2} &\equiv \left(b^i\right)^{(p-1)/2} \bmod p \\ &\equiv b^{i(p-1)/2} \bmod p. \end{aligned}$$

Com que b és d'ordre $p-1$ per ser un element primitiu mòdul p, $p-1 \mid \frac{i(p-1)}{2}$. Açò implica que i ha de ser parell i aleshores, les arrels quadrades de a són $\pm b^{i/2} \bmod p$. □

Així doncs, amb aquest criteri podem determinar si un nombre és un residu quadràtic mòdul p amb una exponenciació que, per mitjà del mètode de l'elevació al quadrat de manera repetida, és una operació d'ordre $O(\log^3 p)$.

Seguidament, presentem el funcionament del mètode de garbell quadràtic:

1. Triar una fita B.
2. Obtindre tots els nombres primers fins a B (es pot usar l'algoritme del garbell d'Erastosthenes).

3. Usant el criteri d'Euler, es pot determinar si el nombre a factoritzar n és un residu quadràtic mòdul cadascun dels nombres primers obtinguts en l'anterior pas. Usant aquest criteri, ens estalviem la necessitat de factoritzar tots els nombres Q_i.

4. Calcular la seqüència de nombres $Q_i = (x+i)^2 - n$ per a $i = 0, 1, 2, 3, ...$ on x és el nombre enter $x \geq \sqrt{n}$ més pròxim a $\sqrt{n}$. Com ja s'ha dit abans, la quantitat de Q_i ha de ser major a la quantitat de nombres primers que es consideren per a factoritzar. Una opció per tal d'assegurar-nos trobar la factorització és sis vegades el tamany de la base factor.

5. Resoldre la congruència $(x+i)^2 - n \equiv 0 \bmod p$ per a cada p de la base factor. Aquest pas es pot realitzar mitjançant l'algoritme Tonelli-Shanks [10]. A més a més, es redueixen els Q_i dividint-los entre els nombre primers $p \in \mathcal{B}$ que els pertoque.

6. Construir una matriu amb els exponents dels factors primers dels Q_i que s'han reduït a 1. Trobar el nucli de la matriu (per a matrius grans són més eficients algoritmes alternatius a la reducció de Gauss com l'algoritme de Lanczos [15] o el de Wiedemann [22]).

7. Amb les solucions que s'han obtingut en l'anterior pas, calcular el $m.c.d.(x+y, n)$ i $m.c.d.(x-y, n)$ amb l'algoritme d'Euclides.

Exemple 12. Tornant a la factorització de 227179, en primer lloc decidiríem el valor de $B = 50$. Seguidament obtindríem tots els nombres primers $\leq B$ que són $\{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 37, 41, 43, 47\}$. Aplicant ara el criteri d'Euler, el tamany de la base es redueix a $\mathcal{B} = \{2, 3, 5, 7, 13, 17, 23, 29, 37, 41, 43, 47\}$.

El següent pas és el de filtrar els nombres generats $(x+i)^2 - n$ per tal de quedar-nos només amb els que es factoritzen completament amb la base factor. Aquest pas es pot realitzar resolent l'equació de congruència $(477+i)^2 - 227179 \equiv 0 \bmod p$, on $p \in \mathcal{B}$. Després d'obtindre $(477+i)^2 - 227179 \equiv 0 \bmod p$ per a cada primer p de la base, dividim els valors corresponents per p. Així, quan després de dividir per un primer p obtenim un 1, voldrà dir que eixa entrada es factoritza de manera completa en la base factor.

Per a $p = 2$, $(477+i)^2 - 227179 \equiv 0 \bmod 2$ dona $i \equiv 0 \bmod 2$, o $i = 2k$ amb $k \in \mathbb{N}$. Així doncs, $i = 0, 2, 4, 6, ...$ És a dir, $(477+i)^2 - 227179$ és divisible entre 2 quan i és parell.

Per tant, si en un principi tenim els següents nombres:

i	0	1	2	3	4	5	6	7	8	9
$(x+i)^2 - 227179$	350	1305	2262	3221	4182	5145	6110	7077	8046	9017

i	10	11	12	13	14	15	16	17
$(x+i)^2 - 227179$	9990	10965	11942	12921	13902	14885	15870	16857

i	18	19
$(x+i)^2 - 227179$	17846	18837

haurem de dividir les entrades parells per 2:

i	0	1	2	3	4	5	6	7	8	9
$(x+i)^2 - 227179$	175	1305	1131	3221	2091	5145	3055	7077	4023	9017

i	10	11	12	13	14	15	16	17
$(x+i)^2 - 227179$	4995	10965	5971	12921	6951	14885	7935	16857

i	18	19
$(x+i)^2 - 227179$	8923	18837

Per a cada $p > 2$, la congruència tindrà dues solucions. Per a $p = 3$, les dues solucions són $i = 3k + 1$ i $i = 3k + 2$. Dividint aquestes entrades entre 3 tenim:

i	0	1	2	3	4	5	6	7	8	9
$(x+i)^2 - 227179$	175	435	377	3221	697	1715	3055	2359	1341	9017

i	10	11	12	13	14	15	16	17
$(x+i)^2 - 227179$	1665	3655	5971	4307	2317	14885	2645	5619

i	18	19
$(x+i)^2 - 227179$	8923	6279

Per a $p = 5$, les solucions són $i = 5k$ i $i = 5k + 1$:

i	0	1	2	3	4	5	6	7	8	9
$(x+i)^2 - 227179$	35	87	377	3221	697	343	611	2359	1341	9017

i	10	11	12	13	14	15	16	17
$(x+i)^2 - 227179$	333	731	5971	4307	2317	2977	529	5619

i	18	19
$(x+i)^2 - 227179$	8923	6279

Continuant amb aquest mateix procediment per a la resta de nombres primers de $\mathcal{B}$, obtenim la següent taula:

i	0	1	2	3	4	5	6	7	8	9
$(x+i)^2 - 227179$	5	3	1	3221	1	49	1	337	1341	9017

i	10	11	12	13	14	15	16	17	18	19
$(x+i)^2 - 227179$	9	1	853	4307	331	229	23	5619	8923	3

Aquest procediment també s'hauria de seguir per a les potències menudes dels nombres primers de la base. Després d'aquest procés, la taula resultant seria la següent.

i	0	1	2	3	4	5	6	7	8	9
$(x+i)^2 - 227179$	1	1	1	3221	1	1	1	337	149	9017

i	10	11	12	13	14	15	16	17	18	19
$(x+i)^2 - 227179$	1	1	853	4307	331	229	1	1873	8923	1

Les entrades de la taula que s'han reduït a 1 són els nombres a considerar per a construir la matriu. A més a més, la seua factorització l'hem aconseguida durant aquest procés. Així doncs, els nombres a considerar junt amb la seua factorització són:

$$\begin{aligned}
350 &= 2 \cdot 5^2 \cdot 7 \\
1305 &= 3^2 \cdot 5 \cdot 29 \\
2262 &= 2 \cdot 3 \cdot 13 \cdot 29 \\
4182 &= 2 \cdot 3 \cdot 17 \cdot 41 \\
5145 &= 3 \cdot 5 \cdot 7^3 \\
6110 &= 2 \cdot 5 \cdot 13 \cdot 47 \\
9990 &= 2 \cdot 3^3 \cdot 5 \cdot 37 \\
10965 &= 3 \cdot 5 \cdot 17 \cdot 43 \\
15870 &= 2 \cdot 3 \cdot 5 \cdot 23^2 \\
18837 &= 3^2 \cdot 7 \cdot 13 \cdot 23.
\end{aligned}$$

Amb aquests nombres construïm com ja hem vist abans la matriu de paritat corresponent als exponents dels nombres primers per a cadascuna de les factoritzacions.

$$\begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}.$$

Calculant el nucli, obtenim la solució $[1,0,0,0,1,0,0,0,1,0]$ que correspon a la mateixa factorització obtinguda anteriorment. Seguint el mateix procediment, obtenim la factorització $n = 1447 \cdot 157$.

La tria de B és molt important per tal d'acurtar el cost computacional d'aquest algoritme alhora que ens permeta factoritzar el nombre n. Un valor adequat determinat

de manera heurística és

$$B = e^{(1/2+o(1))\sqrt{\log n \log\log n}}.$$

El cost computacional dels algoritmes més usats per a factoritzar n són:

Garbell quadràtic	$O\left(e^{(1+o(1))\sqrt{\log n \log\log n}}\right)$
Corba el·líptica	$O\left(e^{(1+o(1))\sqrt{2\log p \log\log p}}\right)$
Garbell sobre el cos de nombres	$O\left(e^{(1.92+o(1))(\log n)^{1/3}(\log\log n)^{2/3}}\right)$

Aquí, $o(1)$ denota una funció que tendeix a 0 quan n tendeix a ∞. Com es pot observar, tots tres algoritmes són de cost exponencial. De manera que no seria factible aplicar-los per a factoritzar n d'un tamany raonable (com 1024 bits).

Altres atacs

Una manera d'atacar el RSA és intentar factoritzar n per tal de poder obtindre $\varphi(n)$. Però una alternativa és directament calcular $\varphi(n)$ sense necessitat de trobar la seva factorització. En aquest cas tindríem:

$$n = pq$$
$$\varphi(n) = (p-1)(q-1).$$

Si substituïm $q = \frac{n}{p}$ de la 1a equació a la 2a obtenim l'equació de segon grau

$$p^2 - (n - \varphi(n) + 1)p + n = 0.$$

Però aleshores les dues arrels serien p i q (que són els factors de n). Per tant, calcular $\varphi(n)$ no és més senzill que factoritzar n.

Atac de Wiener

Una altra opció és el de determinar a, que un altre cop torna a ser equivalent a factoritzar n. Açò és el que realitza l'atac de Wiener i de fet, aconsegueix sota unes condicions. Primer però, cal introduir unes definicions junt amb un resultat que no és demostrarà.

Definició 13. *Una fracció contínua finita és una m-tupla d'enters no negatius $[q_1, q_2, ..., q_m]$ que denota l'expressió:*

$$q_1 + \cfrac{1}{q_2 + \frac{1}{q_3 + \cdots + \frac{1}{q_m}}}.$$

Siguen a i b dos enters positius tals que $m.c.d.(a,b) = 1$ (tal i com en el RSA), siguen $q_1, q_2, ..., q_m$ els quocients que s'obtenen quan s'aplica l'algoritme d'Euclides, aleshores $a/b = [q_1, \ldots, q_m]$. En aquest cas, es diu que $[q_1, \ldots, q_m]$ és l'expansió en fraccions contínues de a/b.

Per a $1 \leq j \leq m$, es defineix $C_j = [q_1, \ldots, q_j]$. C_j s'anomena el j-èssim convergent de $[q_1, \ldots, q_m]$. Cada C_j es pot escriure com un nombre racional c_j/d_j, on cada c_j i d_j compleix la següent relació de recurrència:

$$c_j = \begin{cases} 1 & \text{si } j = 0 \\ q_1 & \text{si } j = 1 \\ q_j c_{j-1} + c_{j-2} & \text{si } j \geq 2 \end{cases},$$

$$d_j = \begin{cases} 0 & \text{si } j = 0 \\ 1 & \text{si } j = 1 \\ q_j d_{j-1} + d_{j-2} & \text{si } j \geq 2 \end{cases}.$$

Teorema 4. *Si* $m.c.d.(a,b) = m.c.d.(c,d) = 1$ *i*

$$\left|\frac{a}{b} - \frac{c}{d}\right| < \frac{1}{2d^2}.$$

Aleshores, c/d *és un dels convergents de l'expansió en fraccions contínues de* a/b.

Aquest atac funciona tal i com demostrarem a continuació si $3a < n^{\frac{1}{4}}$ i $q < p < 2q$. És a dir, aquest atac funcionarà quan la quantitat de bits de a siga menor que $\frac{l}{4} - 1$ on l denota la quantitat de bits de n. Per tant, es pot evitar triant a suficientment gran (malgrat que això implique ralentir quatre vegades més el temps de desencriptar un missatge amb un a sensible a aquest atac).

Si aquestes condicions es compleixen, com que $n = pq > q^2$, tenim que $q < \sqrt{n}$. D'altra banda, $ab \equiv 1 \bmod \varphi(n)$ per definició. Fet que implica que existeix un $t \in \mathbb{Z}$ tal que $ab - t\varphi(n) = 1$. Així,

$$0 < n - \varphi(n) = p + q - 1 < 2q + q - 1 < 3q < 3\sqrt{n}.$$

D'altra banda,

$$\begin{aligned} \left|\frac{b}{n} - \frac{t}{a}\right| &= \left|\frac{ba - tn}{an}\right| \\ &= \left|\frac{1 + t(\varphi(n) - n)}{an}\right| \end{aligned}$$

$$\left|\frac{b}{n}-\frac{t}{a}\right| < \frac{3t\sqrt{n}}{an}$$
$$= \frac{3t}{a\sqrt{n}}.$$

Com que $t < a$, tenim que $3t < 3a < n^{\frac{1}{4}}$, fet pel qual

$$\left|\frac{b}{n}-\frac{t}{a}\right| < \frac{1}{an^{\frac{1}{4}}}.$$

Per acabar, pel fet que $3a < n^{\frac{1}{4}}$, tenim

$$\left|\frac{b}{n}-\frac{t}{a}\right| < \frac{1}{3a^2}.$$

Per tant, la fracció t/a és molt pròxima a la fracció b/n. Pel teorema que hem vist, $\frac{t}{a}$ és un dels convergents de l'expansió en fraccions contínues de b/n. Seguidament, exposem l'algoritme corresponent a aquest atac.

Algoritme Wiener

1. Aplicar l'algoritme d'Euclides per a obtindre els quocients $q_1, q_2, ..., q_m$.
2. Inicialitzar $c_0 = 1$, $c_1 = q_1$, $d_0 = 0$, $d_1 = 1$, $j = 2$.
3. Calcular $c_j = q_j c_{j-1} + c_{j-2}$, $d_j = q_j d_{j-1} + d_{j-2}$ i $n' = (d_j b - 1)/c_j$ (si c_j/d_j és el convergent correcte, $n' = \varphi(n)$).
4. Si n' és enter, resoldre l'equació $x^2 - (n - n' + 1)x + n = 0$, obtenint les arrels p i q. Si aquestes arrels són enters positius menors que n, aleshores retorna p i q (factors de n) i acaba.
5. Incrementa j una unitat.
6. Repetir des del pas 3 fins el 5 fins arribar a $j = m$ inclòs.
7. Retorna "fracàs"en cas que no s'hagen trobat p i q (perquè les condicions inicials $3a < n^{\frac{1}{4}}$ i/o $q < p < 2q$ no es complien).

Exemple 13. Siga $n = 160523347$ i $b = 60728973$. L'expansió en fraccions contínues de b/n és

$$[0, 2, 1, 1, 1, 4, 12, 102, 1, 1, 2, 3, 2, 2, 36].$$

Els primers convergents són $0, \frac{1}{2}, \frac{1}{3}, \frac{2}{5}, \frac{3}{8}, \frac{14}{37}$. Amb els primers 5 primers convergents no arribem a la factorització de n. Però amb $\frac{14}{37}$ sí:

$$n' = \frac{37 \times 60728973 - 1}{14} = 160498000.$$

En aquest cas, si resolem l'equació

$$x^2 - 25348x + 160523347 = 0,$$

obtenim les arrels $x = 12347$ i $x = 13001$. Així, la factorització de n és

$$160523347 = 12347 \times 13001.$$

Així, si $a < \frac{n^{1/4}}{3} \approx 37,52$ podrem desxifrar el missatge.

El mateix Wiener va proporcionar alguns mètodes per a evitar el seu atac al mateix temps que no es ralentia tant el procediment de desencriptació. Una primera opció és triar un $b' = b + \varphi(n)t$ on $t \in \mathbb{Z}$ és un enter gran per a encriptar els missatges. Així, si $b' > n^{\frac{1}{5}}$, l'algoritme no tindrà èxit, però ralentirà el procés d'encriptació. Una segona opció és usar el teorema del residu xinés. Així, si hom tria un a tal que $a_p = a \bmod (p-1)$ i $a_q = a \bmod (q-1)$ són menuts (uns 128 bits cadascun). Aleshores es pot desencriptar de manera ràpida un missatge xifrat C de la següent manera: primer es desencripta el missatge seguint el procediment habitual però per a p i q amb els seus exponents corresponents. És a dir, $M_p \equiv C^{a_p} \bmod p$ i $M_q \equiv C^{a_q} \bmod q$. Tot seguit, s'usa el teorema del residu xinés per a calcular l'únic $M \in \mathbb{Z}_n$ que compleix ambdues equacions: $M \equiv M_p \bmod p$ i $M \equiv M_q \bmod M_q$. Per tant, M satisfà $M \equiv C^a \bmod n$. D'aquesta manera, encara que a_q i a_p siguen menuts, $a \bmod \varphi(n)$ pot ser gran i com a conseqüència, no es puga obtindre un èxit amb aquest algoritme. Tot i així, en aquest cas el temps per a factoritzar n és redueix a $O(min(\sqrt{a_p}, \sqrt{a_q}))$. Per tant, a_p i a_q no poden ser tampoc massa menuts.

6.2.4 Seguretat

Fins ara hem vist que triats els paràmetres n, p, q, a i b de manera adient, el sistema RSA és segur en el sentit que no es factible cap dels atacs coneguts fins ara per tal de desencriptar un missatge encriptat.

Tanmateix, només hem tractat el cas que un mateix missatge s'envia a un únic receptor. A continuació, analitzarem quins altres aspectes cal tenir en compte quan s'envia un mateix missatge a diferents destinataris.

En primer lloc, en cas que vulguem estalviar-nos generar un nombre n amb els seus dos factors primers únics, hom pot pensar que és suficient en usar diferents exponents b i a. Però si tinguérem, posem per cas, dos destinataris cadascú amb la seva clau pública (n, b_1) i (n, b_2) i la seva clau privada (p, q, a_1) i (p, q, a_2) respectivament. A l'hora d'encriptar

un missatge per al primer $C = M^{b_1} \mod n$, el segon el podria desxifrar perquè coneix la factorització de n i l'exponent públic b_1. Per tant, només hauria de trobar l'invers $b_1^{-1} \mod n$ que costa $O(\log^3 n)$. Així doncs, per a cada destinatari diferent, cal un mòdul n diferent.

En segon lloc, en cas d'encriptar el mateix missatge a 3 destinataris o més, no és suficient amb usar un mòdul n diferent per a cadascun. Així, suposem que encriptem el mateix missatge M per a tres destinataris seguint el sistema RSA amb clau pública $(n_i, 3)$ per a $i \in \{1, 2, 3\}$ (per simplicitat, $b = 3$). Aleshores, $C_1 = M^3 \mod n_1$, $C_2 = M^3 \mod n_2$ i $C_3 = M^3 \mod n_3$, serien les tres corresponents encriptacions. Assumint que el $m.c.d.(n_i, n_j) = 1$ per a tot $i \neq j$ (en cas contrari es podria factoritzar algun n_i i desencriptar el missatge), pel teorema del residu xinés tenim que $C' = M^3 \mod n_1 n_2 n_3$. Com que $M < n_i$ per a $1 \leq i \leq 3$, $M^3 < n_1 n_2 n_3$ i per tant, $C' = M^3$. Així, es podria recuperar M calculant l'arrel cúbica de C'. Tot i així, aquest atac només és factible quan els exponents són els mateixos i menuts.

D'altra banda, el sistema RSA per si sol no es considera segur semànticament. És a dir, donada la clau pública (n, b) i un text xifrat C, és possible recuperar part de la informació del text original M. Per tal d'evitar açò, s'afegeix dades abans de xifrar el missatge. Una primera aproximació és donat un missatge M de m bits de longitud, afegir-li dades segons el destinatari i abans d'encriptar el missatge de manera que el missatge a encriptar siga $M_i = i2^m + M$. No obstant això, aquesta manera d'afegir dades no és segura i l'oponent podria arribar a desxifrar el missatge. De fet, qualsevol manera d'afegir dades al missatge original que es puga descriure amb un polinomi no serà segura si hi ha suficients destinataris tal i com observarem a continuació gràcies a Hastad. Però abans, enunciarem el teorema de Coppersmith sense demostració perquè ens serà necessari per a demostrar la generalització del teorema de Hastad.

Teorema 5 (Coppersmith)**.** *Siga N un enter i $g \in \mathbb{Z}[x]$ un polinomi mònic de grau d. Siga $X = N^{\frac{1}{d}-\epsilon}$ per a algun $\epsilon \geq 0$. Aleshores donats (N, g), es poden trobar de manera eficient tots els enters $|x_0| < X$ que satisfan que $g(x_0) = 0 \mod N$.*

Tot seguit, enunciarem i demostrarem la generalització del teorema de Hastad.

Teorema 6. *Siguen $N_1, \ldots N_k$ coprimers dos a dos, $N_{\min} = \min_i(N_i)$ i $g_i \in \mathbb{Z}_{N_i}[x]$ k polinomis de grau com a molt d. Suposem que existeix un únic $M < N_{\min}$ que satisfà que*

$$g_i(M) = 0 \mod N_i \quad \text{per a tot } i = 1, \ldots, k.$$

Si $k > d$, es pot obtindre M donats $(N_i, g_i)_{i=1}^k$.

Demostració. Siga $\bar{N} = N_1 \cdots N_k$. Podem suposar que tots els g_i són mònics. En cas que per a algun i, el coeficient lider no fora invertible en $\mathbb{Z}^*_{N_i}$, es podria obtindre la factorització de N_i. Multiplicant cada g_i per la potència apropiada de x, podem assumir també que tots són de grau d. Llavors construïm el polinomi

$$g(x) = \sum_{i=1}^{k} T_i g_i(x), \quad \text{on } T_i = \begin{array}{ll} 1 \bmod N_j & \text{si } i = j \\ 0 \bmod N_j & \text{si } i \neq j. \end{array}$$

Els T_i s'anomenen els coeficients del residu xinés i es poden obtindre mitjançant aquest. D'aquest polinomi construït observem que és de grau d, és mònic perquè el coeficient lider és 1 mod N_i per a qualsevol i i $g(M) \equiv 0 \bmod \bar{N}$.

Així doncs, podem aplicar el teorema de Coppersmith per a $M < N_{\min} \leq \bar{N}^{1/k} < \bar{N}^{1/d}$ obtenint que es poden trobar tots els enters $|x_0| < \bar{N}^{1/d}$ tal que $g(x_0) \equiv 0 \bmod \bar{N}$. En particular, es pot trobar M tal que $g(M) \equiv 0 \bmod \bar{N}$. □

Amb aquest teorema, queda demostrat que si l'emissor envia a diversos receptors un mateix missatge modificant abans aquest amb una funció polinòmica diferent per a cada receptor, el missatge es pot desencriptar de manera eficient sempre que hi hagen suficients receptors. Per tal d'evitar aquest atac, la manera d'afegir les dades al missatge abans d'encriptar-lo ha de ser aleatòria [6].

6.3 Xifratge homomòrfic

Per a acabar aquest capítol, es realitzarà una breu introducció al xifratge homomòrfic.

La idea de xifratge homomòrfic va néixer un any després del naixement del RSA per part de Ronald Rivest, Leonard Adleman i Michael Dertuzos (els primers dos, inventors del RSA) l'any 1978. Aquest xifratge es defineix com un xifratge que permet realitzar operacions amb les dades xifrades sense necessitat de desxifrar-les abans. Així, el posseïdor de les dades és l'únic qui amb la seva clau privada pot desxifrar-les i obtindre el resultat. Açò és d'especial interès pel que fa a la seguretat dels serveis al núvol, ja que actualment aquesta seguretat es basa en la confiança. De fet, encara que els treballadors d'empreses com Google tenen prohibit accedir a les dades privades, l'any 2010 es va descobrir que un treballador de Google utilitzava els seus privilegis per a accedir a informació privada d'usuaris com per exemple, les sessions de xat. Amb el FHE, aquest incident es podria evitar puix que el servidor no té la possibilitat de veure les dades.

El xifratge homomòrfic es divideix en tres tipus: *partial homomorphic encryption* (PHE), *fully homomorphic encryption* (FHE) i *somewhat homomorphic encryption* (SHE). El primer permet realitzar un tipus d'operació (suma o multiplicació) amb les dades xifrades sense límit pel que fa al nombre de vegades que es pot realitzar aquesta operació. El segon permet els dos tipus d'operacions també sense límit. Per acabar, el tercer permet els dos tipus d'operacions però, només una quantitat limitada de vegades. Sistemes d'encriptació coneguts en el moment del naixement del xifratge homomòrfic ja eren PHE.

Nota 3. El RSA és un PHE per a la multiplicació.

Donats dos missatges M_1 i M_2, desencriptar la multiplicació de les seues encriptacions resulta en la multiplicació dels missatges.

$$\begin{aligned} d(e(M_1) \cdot e(M_2)) &= d(M_1^b \cdot M_2^b \bmod n) \\ &= d(M_1 \cdot M_2)^b \bmod n) \\ &= ((M_1 \cdot M_2)^b)^a \bmod n \\ &= ((M_1 \cdot M_2)^{ba} \bmod n \\ &= M_1 \cdot M_2 \bmod n. \end{aligned}$$

Tanmateix, obtindre un sistema FHE no és tan senzill, i no va ser fins el 2009 quan es va presentar el primer sistema d'encriptació FHE per Craig Gentry [13]. A continuació, no exposarem aquest, sinó que exposarem un altre més senzill: l'algoritme d'aproximació del vector propi. Aquest FHE proposat per Gentry està basat en el problema *Learning With Errors* (LWE) [9]. De fet, avui en dia es pensa que els sistemes basats en aquest problema són segurs fins i tot considerant ordinadors quàntics. Primer enunciarem el problema en què es basa i després descriurem el sistema.

6.3.1 LWE

Per tal de presentar el LWE primer cal presentar el concepte de reticle.

Definició 14. *Siga K un cos, V un espai vectorial de dimensió n sobre K, $B = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ una base de V i R un anell contingut en K. Aleshores el R-reticle $\mathcal{L}$ en V generat per B ve donat per:*

$$\mathcal{L} = \left\{ \sum_{i=1}^{n} a_i \mathbf{v}_i \text{ tal que } a_i \in R \right\}.$$

Per al cas de l'algoritme d'aproximació del vector propi, es treballa en l'anell $\mathbb{Z}_q^n$ i el cos $\mathbb{R}^n$. D'aquesta manera, donada una base $B = \left(\vec{b}_1, \ldots, \vec{b}_n\right) \in \mathbb{R}^{n \times n}$ de vectors linealment independents sobre $\mathbb{R}^n$, el reticle en qüestió està format per totes les combinacions lineals

$$a_1\vec{b}_1 + \cdots + a_n\vec{b}_n \in \mathbb{R}^n$$

on $a_1, \ldots, a_n \in \mathbb{Z}_q$.

El problema de gran dificultat al qual s'aferra la seguretat del sistema d'encriptació proposat per Gentry és el *closest vector problem* (CVP) enunciat a continuació.

Definició 15. *Donat un punt P i un reticle $\mathcal{L}$, el problema del vector més pròxim (Closest Vector Problem) consisteix a trobar el punt L del reticle més pròxim a P.*

Així, el LWE ens demana trobar el punt del reticle $x \in \mathbb{Z}_q^n$ donada una seqüència d'equacions lineals en x. Per tant, en un reticle cal distingir els vectors que s'han creat d'un conjunt d'equacions lineals amb error (soroll) dels vectors uniformement aleatoris. L'error és necessari per tal que trobar x sense cap informació prèvia siga difícil i, per tant, segur. Un dels mètodes per a resoldre aquest sistema de congruències amb error és el mètode de màxima versemblança que requereix de $O(n)$ equacions amb un temps d'execució total de $2^{O(n \log n)}$. Els sistemes d'encriptació homomòrfics basats en el LWE han de mantindre els errors sota un llindar, en cas de no fer-ho la desencriptació pot ser errònia ja que la resolució del CVP pot donar un altre punt més pròxim diferent a l'original. Aquest fet és el causant dels problemes de rendiment que tenen en el present els FHE.

6.3.2 L'algoritme d'aproximació del vector propi

Aquest algoritme permet les operacions de multiplicació i suma en forma de producte i suma de matrius. El sistema tracta d'obtindre el vector propi aproximat $\vec{v}$ tal que siga T una matriu quadrada, $T(\vec{v}) = \lambda\vec{v}$ on λ és un escalar. Per a mantindre l'error, s'aplana la matriu xifrada de manera que continga els valors $\{0, 1\}$.

Per a definir un FHE, en primer lloc es defineixen els paràmetres, després les generacions de la clau privada i de la pública i finalment, els algoritmes d'encriptació i desencriptació. Els paràmetres d'aquest sistema són:

- q: mòdul.

- n: dimensió del reticle.
- $\mathcal{X}$: distribució de l'error.
- $m = O(n \log q)$.
- $l = \lfloor \log_2 q \rfloor + 1$, on $\lfloor x \rfloor$ denota l'enter més gran menor o igual a x. És a dir, l és el nombre de bits amb el qual es pot representar q, i amb això, qualsevol element de $\mathbb{Z}_q$.
- $N = (n+1)l$.

Pel que fa a la generació de la clau privada, el procediment és el següent:

1. Generar $\vec{t} \in \mathbb{Z}_q^n$ aleatori.
2. Calcular $\vec{s} = (1, -t_1, ..., -t_n) \in \mathbb{Z}_q^{n+1}$.
3. Calcular $\vec{v} = Powersof2(\vec{s})$, on $Powersof2(\vec{s})$ ve definit com a $Powersof2(\vec{s}) = (s_0 2^0, s_0 2^1, s_0 2^2, ..., s_0 2^{l-1}, ..., s_{k-1} 2^0, s_{k-1} 2^1, ..., s_{k-1} 2^{l-1})$.

Un cop està generada la clau privada, podem generar la matriu A que fa de clau pública seguint aquests passos:

1. Generar aleatòriament la matriu $B \in \mathbb{Z}_q^{m \times n}$.
2. Generar l'error $\vec{e} \in \mathcal{X}^m$.
3. Calcular $\vec{b} = B \cdot \vec{t} + \vec{e}$.
4. Generar la matriu A de manera que $\vec{b}$ siga la seva primera columna i la resta de columnes siguen la matriu B.

Amb açò ja podem descriure com encriptar un missatge $u \in \mathbb{Z}_q$:

1. Generar una matriu uniforme $R \in \{0,1\}^{N \times m}$.
2. Calcular la matriu xifrada $C = Flatten(u \cdot I_N + BitDecomp(R \cdot A)) \in \mathbb{Z}_q^{N \times N}$, on $BitDecomp(a) = (a_{0,bit(0)}, ..., a_{0,bit(l-1)}, ..., a_{k-1,bit(0)}, ..., a_{k-1,bit(l-1)})$ representa cada coeficient de a descompost en bits, i $Flatten(b) = BitDecomp(BitDecomp^{-1}(b))$.

Finalment, el procés de desencriptació és el següent:

1. Siga i tal que $i < l$ i $q/4 < 2^i < q/2$.
2. Calcular $x_i =< C_i, \vec{v} >$ (el producte escalar de la i-èssima fila de C amb la clau privada $\vec{v}$).
3. El missatge és $u = x_i/2^i$.

Seguidament, veurem que efectivament aquest sistema és FHE. Per a això, cal tenir en compte que $C_n \cdot \vec{v} = u_n \cdot \vec{v} + \overrightarrow{e_n}$ (on $\vec{e}$ és menut). És a dir, el missatge xifrat és el resultat de multiplicar el missatge per la clau privada i sumar-li un error petit. Aquesta mateixa clau és la que s'usa després per a desencriptar, de manera que podem dir que aquest sistema és de clau privada. Al mateix temps però, es pot considerar de clau pública perquè hi ha un altre camí per a xifrar el missatge amb la clau pública.

Així doncs, comencem amb la suma:

$$\begin{aligned}(C_1 + C_2) \cdot \vec{v} &= (C_1 \cdot \vec{v}) + (C_2 \cdot \vec{v}) \\ &= (u_1 \cdot \vec{v} + \overrightarrow{e_1}) + (u_2 \cdot \vec{v} + \overrightarrow{e_2}) \\ &= (u_1 + u_2) \cdot \vec{v} + \overrightarrow{e_1} + \overrightarrow{e_2} \\ &= (u_1 + u_2) \cdot \vec{v} + \vec{e} \quad (\text{on } \vec{e} \text{ és menut}) \ .\end{aligned}$$

De la mateixa manera, per al cas de la multiplicació:

$$\begin{aligned}(C_1 \cdot C_2) \cdot \vec{v} &= C_1 \cdot (u_2 \cdot \vec{v} + \overrightarrow{e_2}) \\ &= (C_1 \cdot \vec{v}) \cdot u_2 + (C_1 \cdot \overrightarrow{e_2}) \\ &= (u_1 \cdot \vec{v} + \overrightarrow{e_1}) \cdot u_2 + (C_1 \cdot \overrightarrow{e_2}) \\ &= u_1 \cdot u_2 \cdot \vec{v} + (u_2 \cdot \overrightarrow{e_1}) + (C_1 \cdot \overrightarrow{e_2}) \\ &= u_1 \cdot u_2 \cdot \vec{v} + \vec{e} \quad (\text{on } \vec{e} \text{ és menut}) \ .\end{aligned}$$

Amb açò, queda demostrat que aquest algoritme proporciona un xifratge FHE.

Hi ha altres exemples de FHE, tots però encara no han aconseguit resoldre el problema de l'eficiència i per tant, encara no s'ha arribat a aplicar de manera extensa a serveis al núvol. Tot i així, l'interès per aquest tema continua amb empreses com Microsoft fent recerca en aquest tema. De fet, l'any 2018 es va publicar el primer estàndard per a FHE.

Referències. Al llarg d'aquest capítol s'ha usat el llibre de Stinson *Cryptography theory and practice* [25], Paar i Pelzl *Understanding cryptography: a textbook for students and practitioners* [21] i el de Katz i Lindell *Introduction to modern cryptography: principles and protocols* [19], per a una visió general de la criptografia així com del sistema RSA. A banda també s'han emprat *Cryptography and network security: principles and practice* de Stallings [24] per a la part dels estàndards de clau privada, l'article *Twenty*

years of attacks on the RSA cryptosystem de Boneh [7] per a part de la secció dels atacs al RSA i *Here's how quadratic sieve factorization Works* per a l'explicació del funcionament del garbell quadràtic [4]. A més a més, per a l'última secció del xifratge homomòrfic s'han usat *New direction in cryptography: homomorphic encryption* de A. B. Levina, V. Y. Kadykov i D. I. Kaplun [1] i el capítol 5 de *The cloud security ecosystem* de M. A. Will i R. K.L. Ko [27].

Capítol 7

Conclusions

En aquest treball, hem vist que la seguretat que proporciona un sistema d'encriptació sempre està condicionat per la impossibilitat (en el sentit de no ser factible) de realitzar el procés de desencriptació sense coneixement de la clau privada. Per al cas del RSA la seguretat ve determinada pel problema de factoritzar un nombre gran i la capacitat computacional dels ordinadors actuals, en el cas del sistema usat per Juli César, la seguretat es basava en l'analfabetisme de la gran majoria de la població i la falta d'estudis suficients sobre llengües. Fins i tot en el cas de l'encriptació FHE, la seguretat es basa en un problema NP-hard. En el moment que a causa d'algun avenç en la resolució del problema, aquesta passa ha ser factible, un sistema d'encriptació deixa de ser segur. Com per exemple, el sistema DES fou segur fins que amb motiu de l'avançament dels ordinadors es podia realitzar un atac amb el mètode de la força bruta. I el sistema de Juli César, no fa la seva funció en l'actualitat com a conseqüència de l'evolució en l'estudi de les llengües. D'altra banda, cal remarcar que un sistema d'encriptació no té aplicació fins que no és factible dur-lo a la pràctica amb els recursos del moment. Així, es va haver de buscar una alternativa al Triple-DES, no perquè no fora segur, sinó perquè era massa exigent computacionalment.

Respecte als tipus de sistemes d'encriptació vists en aquest escrit, el sistema de clau privada sol ser més senzill però té la condició d'haver de disposar d'un canal segur per a comunicar la clau privada a usar entre les parts implicades. En el sistema de clau pública però, aquest aspecte no és necessari pel fet que no hi ha la necessitat de comunicar la clau privada. Així, qualsevol pot xifrar un missatge amb la clau pública però només el destinatari dels missatges té la clau privada per a desencriptar-los. Cada tipus de sistema d'encriptació té les seues aplicacions segons el context. Avui en dia, tant el sistema de clau

privada AES com el sistema de clau pública RSA, s'usen de manera abundant en molts contextos diferents. Pel que fa a l'encriptació homomòrfica, aquesta encriptació permet proporcionar seguretat a un usuari alhora altres entitats al núvol realitzen operacions amb les dades sense en cap moment tindre accés al contingut d'aquestes. En aquest camp encara hi ha bastant feina per fer. Tot i que s'han aconseguit sistemes FHE, la seva aplicació encara resulta poc factible degut a la seva ineficiència computacional.

D'altra banda, cal tenir present que per a l'estudi de la seguretat d'un sistema de clau pública com el RSA, no només s'ha d'estudiar la impossibilitat de desencriptar un únic missatge donat. Sinó que un atacant pot obtenir informació del missatge de diverses maneres, com per exemple provant d'encriptar diferents missatges observant si hi ha cap relació. Per tant, cal fer un estudi rigorós dels possibles atacs que es poden realitzar i la manera d'evitar-los. En el cas del RSA, cal afegir dades de manera aleatòria al missatge abans d'encriptar-lo per tal que siga impossible obtindre qualsevol mena d'informació del missatge o dels paràmetres privats.

En conclusió, perquè un sistema d'encriptació es considere bo, cal que siga factible d'aplicar i difícil de desencriptar tenint en compte tots els diferents atacs que es poden dur a terme.

Bibliografia

[1] V. Y. Kadykov A. B. Levina and D. I. Kaplun. New direction in cryptography: homomorphic encryption. In *2021 International Conference Automatics and Informatics (ICAI)*, pages 234–237, 2021.

[2] P. van Oorschot A. Menezes and S. Vanstone. *Handbook of applied cryptography*. CRC, 1996.

[3] A. G. Agargün and E. M. Özkan. A historical survey of the fundamental theorem of arithmetic. *Historia mathematica*, pages 207–214, 2001.

[4] A. Ayodele. Here's how quadratic sieve factorization works. *Medium*, 2022.

[5] P. Bachmann. *Analytische Zahlentheorie*, volume 2. Leipzig: B. G. Teubner, 1894.

[6] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *Advances in cryptology — EUROCRYPT'94*, volume 950, 1995.

[7] D. Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the American Mathematical Society*, 46(2):203–213, 1999.

[8] D. M. Burton. *The history of mathematics / An introduction*. McGraw-Hill, 2011.

[9] A. Sahai C. Gentry and B. Waters. Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in cryptology – CRYPTO 2013*, pages 75–92. Springer Berlin Heidelberg, 2013.

[10] H. Cohen. *A course in computational algebraic number theory*. Springer, 1993.

[11] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6), 1976.

[12] C. F. Gauss. *Disquisitiones arithmeticae*. 1801.

[13] C. Gentry. *A fully homomorphic encryption scheme*. Stanford university, 2009.

[14] G. H. Hardy and E. M. Wright. *An introduction to the theory of numbers*. Oxford University Press, 1979.

[15] R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge University Press, 1989.

[16] G. A. Jones i J. M. Jones. *Elementary number theory*. Springer, 1998.

[17] Melis Jackob. History of encryption. Technical report, SANS Institute, 2021.

[18] Qin Jiushao's. *Mathematical treatise in nine sections*. 1852.

[19] J. Katz and Y. Lindell. *Introduction to modern cryptography: principles and protocols*. Chapman & Hall/CRC, 2015.

[20] N. Koblitz. *A course in number theory and cryptography*. Springer, 1994.

[21] C. Paar and J. Pelzl. *Understanding cryptography: a textbook for students and practitioners*. Springer, 2014.

[22] R. Pass and A. Shelat. A course in cryptography. `https://www.cs.cornell.edu/courses/cs4830/2010fa/lecnotes.pdf`, 2010. Accedit: 2023-09-30.

[23] U. Singh. The square root algorithm. *Medium*, 2021.

[24] W. Stallings. *Cryptography and network security: principles and practice*. Pearson, 2017.

[25] D. R. Stinson. *Cryptography theory and practice*. Chapman & Hall/CRC. Wiley-Interscience, 2006.

[26] Sunzi. *Sunzi Suanjing*. s. III.

[27] Mark A. Will and R. K.L. Ko. Chapter 5 - a guide to homomorphic encryption. In R. Ko and K. R. Choo, editors, *The cloud security ecosystem*. Syngress, 2015.