Anàlisi de la influència de la supercomputació al desxiframent de contrasenyes d'arxius ".zip"

Autor: Alejandro Delgado Estudillo
Tutor: Carlos Sicília Espín
Centre: Sant Antoni de Pàdua
Assessorament: Carlos García Calatrava

Abstract

In the digital era, passwords remain one of the primary mechanisms for protecting sensitive information. Despite advances in security systems, brute-force attacks continue to pose a tangible threat. This research investigates the impact of parallelization on the efficiency of brute-force attacks against encrypted ZIP files, comparing three scenarios (sequential execution, limited-core parallelization on a personal computer and massive parallelization on a supercomputer, specifically Marenostrum 5) and 12 types of passwords, depending on the length (4, 5, and 6 characters) and the charset used (lowercase, lowercase and uppercase, letters and numbers and letters, numbers and symbols),. Custom Python scripts were developed to perform brute-force operations, and experimental measurements focused on the number of password combinations processed per second. Results show that while individual processes in a parallelized environment may operate at lower speeds than sequential execution, the overall performance increases substantially as the number of concurrent processes rises. In particular, massive parallelization enables a significant reduction in the estimated time required to crack passwords, especially those of moderate length. However, password strength remains primarily determined by length and character diversity, with longer and more complex passwords exhibiting high resistance even under supercomputing conditions. Furthermore, the study highlights that access to supercomputing resources is controlled and limited, making large-scale attacks impractical for ordinary users. These findings reinforce the importance of creating robust passwords and employing advanced encryption methods to secure sensitive data. Overall, this study combines theoretical analysis and practical experimentation to provide a comprehensive understanding of how computational power and parallel processing influence password security and cybersecurity strategies.

Índex

Introducció al treball

Vivim en una era en què la informació digital constitueix un dels actius més valuosos. Cada dia, milions de persones i organitzacions protegeixen les seves dades amb contrasenyes, però, malgrat les millores en els sistemes de seguretat, els atacs de força bruta continuen sent una amenaça real i efectiva. Què tan segures són realment les nostres contrasenyes davant de la potència de la supercomputació? Aquesta és la pregunta central que guia aquest treball de recerca.

L'estudi se centra en analitzar com la paral·lelització de programes, de l'ús de nuclis limitats en un ordinador personal fins a la paral·lelització massiva en un superordinador com el Marenostrum 5, afecta l'eficiència dels atacs de força bruta sobre fitxers ZIP xifrats. L'interès d'aquest estudi està en la seva rellevància pràctica i acadèmica: malgrat que els superordinadors com el Marenostrum tenen un accés controlat, ha sigut possible accedir-hi a través de la col·laboració del BSC i l'assessorament d'en Carlos García Calatrava. A més entendre com la paral·lelització pot accelerar el trencament de contrasenyes dona una perspectiva valuosa sobre la vulnerabilitat de dispositius comuns i la necessitat de crear contrasenyes robustes.

Investigacions anteriors han analitzat els mètodes de xifrat i les tècniques de descodificació, però aquest treball aporta una aproximació experimental directa que compara tres escenaris diferents de força bruta.

L'objectiu principal és quantificar l'impacte de la paral·lelització en el temps necessari per desxifrar contrasenyes i establir quines variables (longitud, i grup de caràcters) influeixen més en la seva seguretat. Per aconseguir-ho, s'han desenvolupat codis en Python per aplicar la força bruta tant en un ordinador personal com en un superordinador.

Les proves realitzades mostren que, tot i que la paral·lelització massiva accelera significativament el procés de força bruta, la resistència d'una contrasenya depèn fonamentalment de la seva longitud i complexitat. I que avui en dia, l'accés a superordinadors per realitzar atacs d'aquest tipus és inviable. Aquestes evidències reforcen la importància de crear contrasenyes robustes i utilitzar sistemes de xifrat avançats per protegir la informació.

En resum, aquest treball de recerca combina teoria i pràctica per analitzar un problema de gran rellevància actual: la seguretat de la informació davant de les tècniques de descodificació amb capacitat computacional elevada. Els resultats obtinguts ofereixen conclusions clares sobre els factors que determinen la resistència de les contrasenyes i contribueixen a la comprensió de com la supercomputació pot influir en la ciberseguretat.

1. Teoria necessària i context

1.1 Ciberseguretat i contrasenyes

Per tal d'assolir aquest objectiu de manera efectiva, cal entendre alguns conceptes bàsics de l'àmbit de la ciberseguretat que permeten comprendre com funcionen els mètodes de desxiframent i altres idees relacionades.

Comencem amb les bases. Com s'explica al glossari de la NICCS (2025) i al llibre *Computer Security: art and science (Bishop, 2018) ,* els actius informàtics (*assets*) són tot allò que té valor per al seu propietari: dades, dispositius, informació, arxius, etc. Aquests actius poden tenir vulnerabilitats, que són debilitats o punts febles que poden ser explotats. Una amenaça (*threat*) és una possible acció que pot aprofitar una vulnerabilitat per causar un dany. Els atacants aprofiten aquestes vulnerabilitats mitjançant tècniques anomenades *exploits*, i així fan realitat l'amenaça potencial.

Dins de l'àmbit dels atacs a claus de seguretat, hi ha una relació directa entre aquests conceptes i els elements que intervenen en els processos de desxiframent. Les contrasenyes son un tipus d'actiu digital que, sovint, presenten vulnerabilitats (com ara ser massa simples, previsibles o reutilitzades). Aquestes debilitats poden ser aprofitades per una amenaça (com un atacant), que utilitza tècniques específiques per explotar-les. Aquests mètodes, coneguts com a tècniques de desxiframent o password cracking, són l'eina per la qual l'amenaça es fa real. A continuació, s'expliquen els més habituals.

1.1.1 Mètodes de desxiframent de contrasenyes

Quan es tracta de garantir la seguretat digital, és vital entendre com els ciberdelinqüents ataquen les contrasenyes. Els mètodes de desxiframent, també coneguts com a tècniques de *cracking*, han anat evolucionant amb el pas del temps i aprofitant l'avenç de la potència computacional. D'entre aquests mètodes en destaquen els més utilitzats i explicats a l'entrada al blog de BeyondTrust *Password Cracking 101* de Matt Miller (2024), que són:

a) Força bruta

Un atac de força bruta consisteix a provar totes les combinacions possibles d'una contrasenya fins a encertar-la (Miller, 2024).

Com menys llarga i complexa sigui, més fàcil serà desxifrar-la. Aquest mètode garanteix trobar la contrasenya tard o d'hora, però pot requerir molt de temps i molta potència computacional, tal com assenyala CAPEC (2023). Per això, és una tècnica poc eficient i moltes vegades es considera l'última opció per als atacants.

Tot i així, en entorns amb una gran capacitat de càlcul, com els superordinadors, aquest mètode es pot aprofitar molt més.

b) Atac de diccionari

Els atacs de diccionari utilitzen llistes automàtiques de paraules habituals per provar contrasenyes (CAPEC, 2023). Sovint combinen aquestes paraules amb números o símbols per imitar contrasenyes reals que compleixen els requisits de complexitat que s'hi posen al crear la clau de seguretat.

És una tècnica més eficient que la força bruta, però no et garanteix encertar la contrasenya al 100% (Miller, 2024). Aquest mètode funciona pitjor si la contrasenya és molt complexa, inventada, o utilitza combinacions poc habituals.

c) Credential Stuffing

El credential stuffing és una tècnica que utilitza informacions robades, com usuaris i contrasenyes (Miller, 2024). Els atacants fan servir aquestes credencials per provar d'entrar en aplicacions i detectar on els usuaris han reutilitzat les mateixes credencials.

Aquest mètode no intenta endevinar contrasenyes, sinó que fa servir combinacions ja conegudes per entrar en altres comptes (CAPEC, 2023). El gran problema és que molts usuaris reutilitzen les mateixes contrasenyes en diferents serveis i no activen sistemes d'autenticació multifactor (MFA), cosa que facilita l'èxit de l'atac.

d) Password Spraying

El password spraying és un atac que intenta entrar a molts comptes provant algunes contrasenyes molt utilitzades, en lloc de moltes combinacions en un sol compte com en la força bruta. (Miller, 2024)

Tal com assenyala CAPEC (2023), l'atacant prova una contrasenya habitual, com "12345678", a tots els comptes abans de passar a una altra contrasenya, així s'eviten bloquejos i detecció. Aquesta tècnica s'aprofita de que molts usuaris fan servir les mateixes contrasenyes febles en molts comptes.

e) Taula Comparativa

Amb aquesta taula es veu clarament les diferències i funcionament de cada mètode, descrivint els avantatges i desavantatges.

Mètode	Funcionament	Pros	Contres
Força bruta	Prova totes les combinacions possibles	Garanteix trobar la contrasenya	Molt lent i costós en temps i potència, especialment per contrasenyes complexes
Atac de diccionari	Utilitza llistes de paraules habituals i combinacions comunes	Més ràpid que la força bruta, fàcil d'automatitzar	No serveix amb contrasenyes inventades o molt complexes
Credential Stuffing	Fa servir credencials robades en altres llocs	Molt eficient si l'usuari reutilitza contrasenyes	Depèn de dades prèviament filtrades; no serveix si es fa servir MFA
Password Spraying	Prova poques contrasenyes comunes en molts comptes	Difícil de detectar, evita bloquejos automàtics	Poc efectiu si els usuaris utilitzen contrasenyes fortes i variades

Figura 1, Taula comparativa dels mètodes de desxiframent de contrasenyes (Elaboració pròpia a partir de dades extretes del CAPEC, 2023 i Miller, 2024)

De tots aquests mètodes, el que té més rellevància per la seva adequació i capacitat demostrativa en aquest treball és el de la força bruta. Això és perquè, com ja s'ha mencionat en la seva respectiva explicació, en casos on la potència computacional és molt elevada, aquest mètode és encara més efectiu i es pot aprofitar.

1.1.2 Sistemes de protecció addicionals

Encara que les contrasenyes són molt importants per protegir la informació, moltes vegades no són suficients per evitar un atac. Per això, existeixen sistemes de seguretat extra que ajuden a protegir millor els comptes, encara que algú descobreix la contrasenya.

Alguns d'aquests mètodes són:

a) Autenticació multifactor (MFA)

Com bé explica Microsoft a l'entrada al seu article web *What is: Multifactor Authentication,* l'autenticació multifactor és un sistema que obliga a verificar la identitat de l'usuari utilitzant més d'un mètode. Normalment, es combinen una contrasenya, algún dispositiu extern, com el teu telèfon, dades biomètriques, com la teva empremta, o preguntes personals. Això fa que encari que algú la contrasenya atacada, no pugui accedir al compte si no té els altres factors.

És un dels mètodes més efectius per evitar que entrin al compte o accedeixin a dades sense permís.

Per exemple, si l'usuari vol entrar al correu des d'un ordinador nou, el sistema pot demanar la contrasenya i després un codi que arriba al mòbil. Sense aquest segon pas, no es pot accedir al compte, encara que la contrasenya sigui correcta. Aquest sistema és cada vegada més comú, sobretot en aplicacions bancàries, correus electrònics i plataformes amb dades confidencials.

b) Gestors de contrasenyes

Un gestor de contrasenyes és una eina que guarda totes les contrasenyes de manera segura i en un sol lloc. Això fa que no s'hagi de recordar moltes contrasenyes o que es faci servir la mateixa en molts llocs, cosa que és molt perillosa.

Els gestors generen contrasenyes fortes i úniques per a cada compte i les guarden de manera xifrada i segura. Quan es vol accedir a un compte o a algún servei, el gestor posa la contrasenya de manera automàtica.

A més a més, molts gestors estan sincronitzats i avisen si alguna de les contrasenyes ha estat compromesa o desxifrada, així es pot canviar-la abans que hi hagi conseqüències majors.

Tot això està molt ben explicat en l'apartat de "Gestor de contrasenyes" de la web de l'empresa Norton (2025).

c) Sistemes de detecció d'activitat sospitosa

Com s'esmenta a l'apartat "What is a WAF" de la web de Cloudflare, i reafirmat per la OWASP, els sistemes de detecció d'activitat sospitosa miren el comportament dels usuaris per identificar intents d'atacs. Fan servir tècniques que reconeixen patrons inusuals com múltiples intents fallits o intents massius des de la mateixa IP.

Un element molt important és els tallafocs d'aplicacions web (WAF), que filtra i bloqueja el trànsit perillós abans que arribi al servidor. Això protegeix contra atacs de força bruta, diccionari i altres amenaces que són automàtiques.

Aquest tipus de sistemes són molt importants per detectar i prevenir atacs i mantenir la seguretat de les aplicacions web.

1.2 Supercomputació

Des del 1964 que la tecnologia ha donat la possibilitat d'aspirar a una capacitat de càlcul major a través de la supercomputació. Aquests tipus de màquina ha donat pas a molts avenços en molts camps diferents. Aquesta polivalència d'aplicacions es pot veure plasmada en les sessions del Bojos per la Supercomputació, que s'han centrat en 4 tipus de branques (les que divideix el BSC):

· *Computer sciences:* amb sessions com "Paral·lelisme" d'en Xavier Martorell o "Infraestructures de dades" d'en Carlos Garcia.

· *Life Sciences:* amb sessions com "Genòmica" d'en Moisès Bernabeu o "Proteïnes i disseny de fàrmacs" de la Júlia Vilalta i l'Isaac Filella.

· *Earth Sciences:* amb sessions com "La modelització del temps i el clima" de la Valentina Sicardi i la Ana Cristina Franco o "Simular la terra" d'en Josep de la Puente

· *Computer Applications in Science and Engineering (CASE):* amb sessions com "Simulant el pas del temps en un laboratori virtual" d'en Xevi Roca o "La intel·ligència artificial aplicada al llenguatge natural" de la Valle Ruiz Fernández i l'Adrián Rubio Pintado

1.2.1 Arquitectura dels sistemes d'HPC

Els sistemes d'HPC (*High Performance Computing*), coneguts com a superordinadors, no només es diferencien dels ordinadors convencionals per la seva gran potència de càlcul, sinó també per la seva arquitectura.

Un clúster HPC és un conjunt de nodes interconnectats que treballen junts per resoldre tasques complexes de càlcul de manera molt eficient. Cada node és com un petit ordinador, format per la seva pròpia CPU (amb un o més nuclis) i memòria RAM.

Aquests nodes no treballen individualment, sinó que treballen de manera coordinada amb un sistema de memòria d'alta velocitat que comparteixen entre tots.

(BSC, 2025)

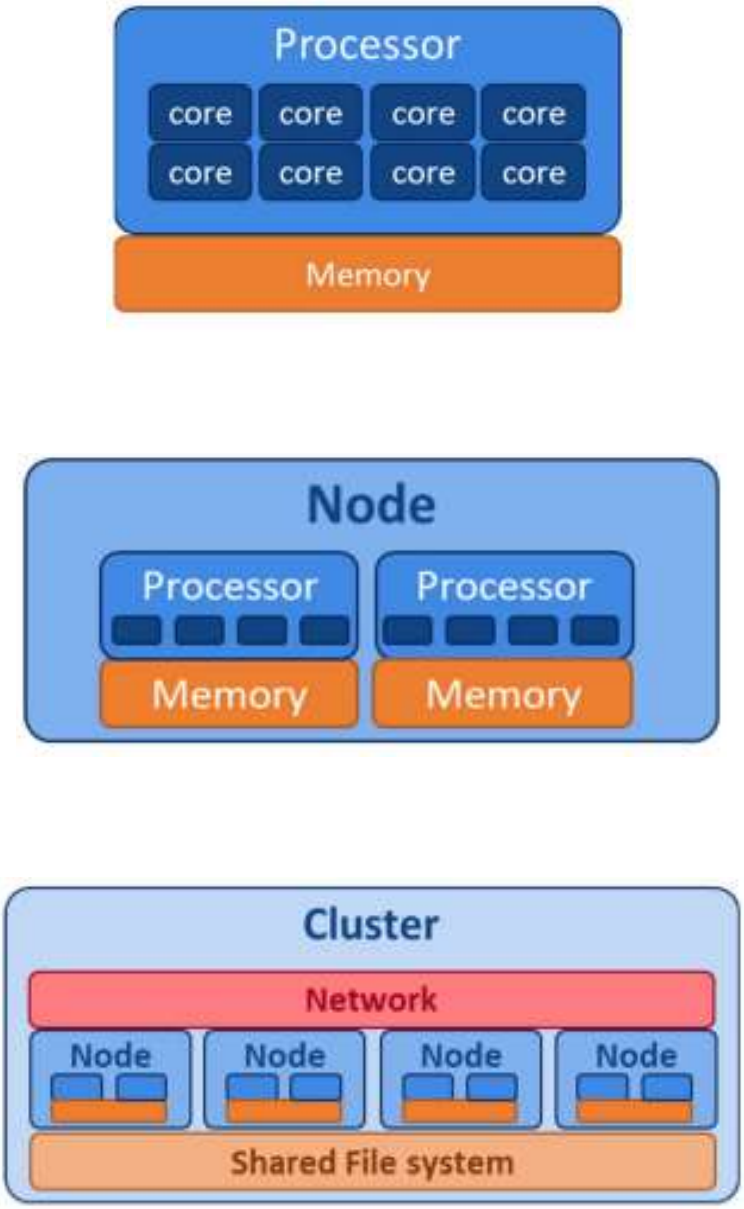

Figura 2, Diagrames de l'arquitectura d'un Superordinador
(BSC , 2025)

1.2.2 Paral·lelisme

El paral·lelisme és la representació directa de la frase cèlebre "Divide et Impera" (divideix i venceràs) de Juli Cèsar al món de la informàtica. Però, per entendre el paral·lelisme primer hem d'entendre altres conceptes, com la llei de Moore:

a) Llei de Moore

La **Llei de Moore** té una implicació directa en l'aparició del paral·lelisme. Com es veu a les presentacions de la UPC (2013) i del BSC (2025), aquesta llei estableix que, cada dos anys aproximadament, el nombre de transistors que es poden posar dins d'un processador es duplica. Això va permetre augmentar la potència dels processadors simplement incrementant la seva freqüència i complexitat. Però, al voltant dels anys 2000, aquesta llei va començar a fallar, ja que l'augment de la freqüència implicava un major consum energètic i problemes tèrmics, fins al punt que ja no era viable continuar per aquest camí. Per tant, la solució no va ser fer processadors més ràpids, sinó afegir-ne més: més nuclis dins d'un mateix xip, capaços de treballar a la vegada de manera coordinada.

b) Execució Serial

Per entendre bé com funciona el paral·lelisme, hem d'entendre que és l'execució serial. Tradicionalment, els programes s'han escrit per executar-se de manera seqüencial, fets per funcionar en un únic processador. Això vol dir que el programa està format per una sèrie d'instruccions que s'executen una rere l'altra, sense que n'hi hagi cap executant-se al mateix temps. (UPC, 2013)

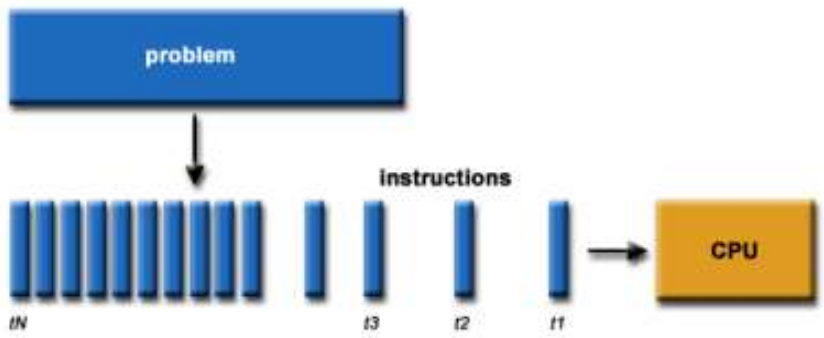

Figura 3, Diagrama esquemàtic de l'execució serial
(Presentacions assignatura PAR, UPC)

c) Concurrència

La **concurrència** implica descompondre un problema en diverses tasques més petites i coordinar la seva execució de manera que el sistema pugui gestionar-les de forma eficient. Com es veu a les presentacions de la UPC (2013) i del BSC (2025), encara que cada tasca s'executa de manera individual, el processador alterna la seva execució, fent semblar que múltiples tasques s'estan realitzant al mateix temps. Aquesta tècnica permet gestionar diverses operacions simultàniament, fins i tot amb un sol nucli.

d) Paral·lelisme

Finalment, el **paral·lelisme** consisteix a utilitzar diversos processadors alhora per executar diferents parts d'un mateix programa. És una evolució de la concurrència, ja que les tasques s'executen al mateix temps en processadors diferents. Segons les presentacions de la UPC (2013) i del BSC (2025), en un cas ideal, si es tenen p processadors, cadascun s'encarregaria d'1/p del total, reduint així el temps d'execució global en un factor p.

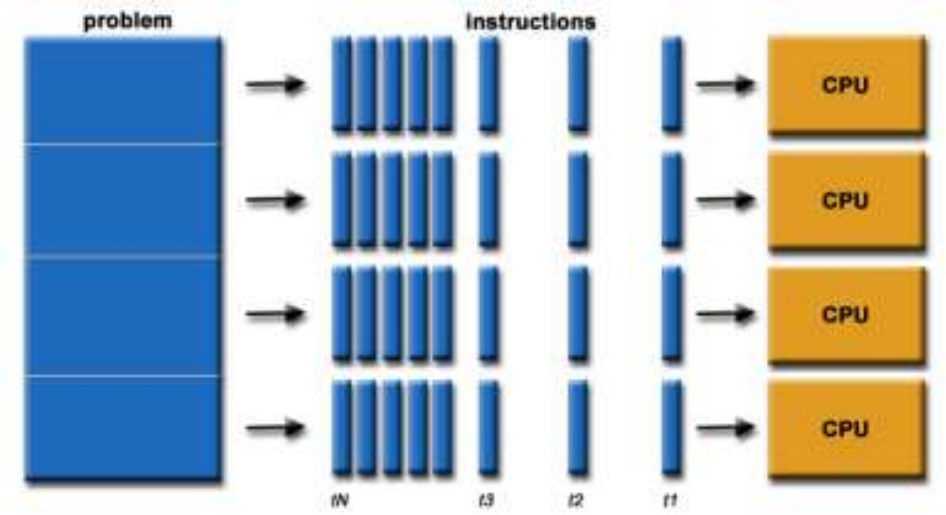

Figura 4, Diagrama esquemàtic del paral·lelisme
(Presentacions assignatura PAR, UPC)

1.2.3 Gestió del paral·lelisme

Encara que el paral·lelisme pot semblar fàcil i simple a primera ullada, aquestes tècniques requereixen d'una planificació i un disseny meticulós per tal d'optimitzar al màxim les execucions i aprofitar de manera eficient tots els recursos disponibles. La configuració del paral·lelisme varia en funció del context i de les característiques del sistema, adaptant-se a cada situació per maximitzar el rendiment. Una implementació correcta permet millorar significativament l'eficiència de l'execució.

a) Sincronia i asincronia

Com bé es va explicar en les sessions amb Carlos Garcia Calatrava al BSC, coordinar les tasques en paral·lel és molt important per aprofitar al màxim els recursos i evitar problemes. Hi ha dues formes principals de fer-ho: la sincronia i l'asincronia.

La sincronització fa que una tasca només pugui continuar quan les altres hagin acabat. Això ajuda a mantenir tot en ordre, però si s'abusa pot fer que el sistema vagi més lent.

L'asincronització, en canvi, permet que les tasques s'executin al mateix temps sense haver d'esperar-se unes a altres. Això pot fer que tot vagi més ràpid, però cal tenir cura de no provocar errors o conflictes amb les dades.

b) Divisió de paquets de treball

Segons les explicacions de Carlos Garcia Calatrava al BSC, dividir les tasques en paquets de treball és essencial per organitzar el paral·lelisme i aprofitar al màxim els recursos disponibles. Cada paquet conté una part del treball total, que pot ser processada de manera independent. Aquesta divisió permet assignar les tasques a diferents recursos simultàniament, millorant així el rendiment global del sistema. És important planificar com es creen i es reparteixen els paquets, ja que una distribució desequilibrada pot fer que alguns recursos estiguin sobrecarregats mentre altres estan inactius.

c) Taula comparativa de configuracions

Configuració	Avantatges	Inconvenients
Síncron i pocs paquets	Molt de control, fàcil de gestionar i depurar.	Pot generar colls d'ampolla i un ús ineficient dels recursos.
Síncron i molts paquets	Manté l'ordre i la coherència de les dades amb moltes tasques.	Molta espera, pot reduir significativament el rendiment.
Asíncron i pocs paquets	Millor ús dels recursos que amb sincronia; menys temps d'espera.	Poden aparèixer conflictes amb les dades; menys control sobre l'ordre d'execució.
Asíncron i molts paquets	Màxim aprofitament dels recursos. Augment del rendiment en entorns amb molta paral·lelització.	Gestió més complexa; risc major de conflictes i errors si no es controla bé.

Figura 5, Taula d'avantatges i inconvenients entre tipus de configuracions (Elaboració pròpia amb informació extreta de les sessions al BSC)

1.3 Situació actual de l'ús de la supercomputació en el desxiframent de contrasenyes

Amb l'avenç constant de la potència computacional, la supercomputació s'ha convertit en una eina clau tant per a la investigació en ciberseguretat com per al desxiframent de contrasenyes en entorns controlats o d'atac real. L'ús de sistemes amb molts nuclis o amb GPUs paral·leles ha permès multiplicar exponencialment la velocitat amb què es poden provar combinacions de contrasenyes, especialment en atacs de força bruta i de diccionari.

Un d'aquests estudis és *"Password Cracking with Brute Force Algorithm and Dictionary Attack Using Parallel Programming"* (Alkhwaja et al., 2023), en què els autors van implementar atacs de força bruta i de diccionari utilitzant programació paral·lela amb OpenMP. Els resultats van mostrar una millora substancial en el rendiment quan es comparaven amb implementacions seqüencials, arribant a multiplicar per gairebé cinc la velocitat d'execució. Aquesta recerca posa de manifest com l'ús de múltiples fils d'execució pot reduir de manera significativa el temps necessari per trencar contrasenyes, sobretot en sistemes amb poca complexitat.

En una línea similar està l'estudi *"A New Distributed Brute-Force Password Cracking Technique"* (Tirado et al., 2022), que utilitza una tècnica que aprofita la distribució de la càrrega computacional entre diferents nodes per augmentar l'eficiència global.. A diferència de les tècniques tradicionals que requereixen un únic superordinador, aquest enfocament permet aprofitar recursos computacionals disponibles de forma descentralitzada.

A diferència dels estudis esmentats, aquest treball no només analitza tècniques generals de desxiframent paral·lel, sinó que aplica aquests coneixements en el context d'un superordinador real com el MareNostrum 5. Aquesta aproximació permet observar com es tradueixen els conceptes teòrics de paral·lelisme en un entorn de supercomputació de primer nivell, establint les bases sobre si aquests tipus d'atacs directes son una amenaça real i com es podria evitar.

2. Anàlisi Matemàtica Contrasenyes

2.1. Introducció de l'anàlisi

Aquest apartat presenta una anàlisi matemàtic de la seguretat de diferents tipus de contrasenyes segons dues variables: la llargada i el tipus de caràcters . Es calcula el nombre de combinacions possibles en cada cas, amb l'objectiu d'avaluar la complexitat teòrica de cada tipus de contrasenya.

Aquesta anàlisi permet comparar la dificultat teòrica de desxifratge de les contrasenyes segons la seva composició i longitud, així com preveure el temps necessari per un atac de força bruta.

2.2. Variables de les contrasenyes

Per a l'anàlisi matemàtica es consideren dues variables principals que condicionen la seguretat de les contrasenyes:

- Llargada**:** Es refereix al nombre total de caràcters que conté la contrasenya. En aquest estudi, es treballa amb tres llargades diferents: 4, 5 i 6 caràcters.

- Grup de caràcters**:** Es refereix al conjunt de caràcters que poden formar la contrasenya. S'han definit quatre categories:

    1. Només minúscules (26 caràcters)
    2. Només números (10 caràcters)
    3. Minúscules i majúscules (52 caràcters)
    4. Minúscules, majúscules i números (62 caràcters)
    5. Minúscules, majúscules, números i símbols especials (94 caràcters)

    Per comprendre millor aquesta variable i com afecta al treball hem de definir bé que inclou cada grup.

En aquest treball, s'han exclòs caràcters com les lletres accentuades (á, é, í, ò, ç, etc.) o símbols com ç o ñ, ja que no formen part del conjunt ASCII estàndard, que és el que utilitzen habitualment els algorismes de generació i trencament de contrasenyes.

Tenint això en compte, els grups queden definits com:

Minúscules (26 caràcters): a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z

Majúscules (26 caràcters): A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z

Números (10 caràcters): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Símbols especials (32 caràcters): !, ", #, $, %, &, ', (,), *, +, ,, -, ., /, :, ;, <, =, >, ?, @, [, \,], ^, _, `, {, |, }, ~

Aquestes variables permeten crear un conjunt de casos que cobreixen diferents nivells de complexitat, des de contrasenyes senzilles fins a contrasenyes més fortes i difícils de desxifrar.

2.3 Procediment matemàtic

Per calcular el nombre total de combinacions possibles s'han d'utilitzar les tècniques de recompte. Hi ha tres tipus, permutacions, combinacions i variacions (amb repetició o sense).

· Una permutació sense repetició és una ordenació de tots els elements d'un conjunt. En aquesta tècnica l'ordre és rellevant i els elements no es poden repetir.

La fórmula general per calcular el nombre total de combinacions és:

$$N = n!$$

Exemple: amb les lletres A, B, C possibles permutacions: ABC, ACB, BAC, BCA, CAB, CBA. En total, 6 combinacions diferents, és a dir, 3! (3! = 3 · 2 · 1).

· Una combinació és una tècnica on es calculen les configuracions possibles d'una selecció d'elements d'un conjunt on no es permeten repeticions i l'ordre en què es trien no importa. Això significa que només importa quins elements formen part del grup, però no l'ordre o la posició que ocupen dins d'aquest grup.

La fórmula general per calcular el nombre total de combinacions és:

$$N = \frac{n!}{k! \cdot (n-k)!}$$

On "n" és el nombre total d'elements del conjunt i "k" és el nombre d'elements que es volen triar.

Exemple: Amb el conjunt de lletres A, B, C, si se'n volen triar 2, les combinacions possibles són: AB, AC i BC.
Les combinacions BA, CA o CB no es compten com a diferents, ja que l'ordre no altera el resultat.

· Una variació sense repetició és una tècnica on es calculen les possibles configuracions de seleccions d'elements d'un conjunt, on l'ordre importa i no es permeten repeticions. És a dir, la posició dels elements sí que és rellevant i no es poden repetir dins la selecció.

La fórmula general per calcular el nombre total de combinacions és:

$$N = \frac{n!}{(n-k)!}$$

On n és el nombre total d'elements disponibles i k és la mida de la selecció.

Exemple: Amb el conjunt de lletres A, B, C, si se'n volen triar 2 i l'ordre importa, les variacions possibles són: AB, AC, BA, BC, CA, CB.

· Una variació amb repetició és una tècnica on es calculen les configuracions possibles de seleccions d'elements d'un conjunt, on l'ordre importa però es permeten repeticions. Això vol dir que la posició dels elements és important i els elements es poden repetir dins la selecció.

La fórmula general per calcular el nombre total de combinacions és:

$$N = n^k$$

On n és el nombre total d'elements disponibles i k la mida de la selecció.

Exemple: Amb el conjunt de lletres A, B, C, si es volen formar cadenes de 2 lletres on l'ordre importa i es permeten repeticions, les variacions possibles són: AA, AB, AC, BA, BB, BC, CA, CB, CC.

Per al càlcul del nombre total de possibles contrasenyes en aquest treball, s'ha escollit modelar el problema utilitzant les variacions amb repetició. Si es volgués fer de manera totalment precisa, s'hauria d'utilitzar aquesta variant:

$$N = \sum_{k\,min}^{k\,max} n^k$$

Aquesta variant contempla que la longitud no es fixa, cosa que a nivell teòric és més correcte. Malgrat això, no es fa ús d'aquesta fórmula ja que computacionalment no té sentit degut al creixement exponencial de la complexitat. Això deriva a que la diferència que hi ha quan es contempla una longitud fixa (agafant la longitud com a la màxima) és minúscula.

1. L'ordre importa:
   En una contrasenya, la posició de cada caràcter és important. Per exemple, la contrasenya abc és diferent de cba.

2. Es permeten repeticions:
   Els caràcters poden repetir-se en la mateixa contrasenya. Per exemple, aaa o 1122 són combinacions vàlides.

3. Longitud fixa (computacionalment) i conjunt definit:
   La contrasenya té una longitud determinada (k) i es forma a partir d'un conjunt limitat de caràcters (n).

Per tant, la situació correspon exactament a una variació amb repetició. Aquesta tècnica reflecteix el creixement exponencial del nombre de contrasenyes possibles quan augmentem la longitud o la varietat de caràcters, i és necessària per avaluar la resistència davant atacs de força bruta o per diccionari.

2.4 Càlculs i resultats

En aquesta taula veiem el nombre de combinacions possibles depenent de les dues variables (Llargada en l'eix vertical; Tipus de caràcters en l'eix horitzontal).

	Minúscules (26 caràcters)	Minúscules i Majúscules (52 caràcters)	Lletres i números (62 caràcters)	Lletres, números i símbols (94 caràcters)
4 caràcters	$3{,}09 \cdot 10^{8}$	$1{,}98 \cdot 10^{10}$	$5{,}68 \cdot 10^{10}$	$6{,}90 \cdot 10^{11}$
5 caràcters	$2{,}09 \cdot 10^{11}$	$5{,}35 \cdot 10^{13}$	$2{,}18 \cdot 10^{14}$	$6{,}10 \cdot 10^{15}$
6 caràcters	$1{,}41 \cdot 10^{14}$	$1{,}45 \cdot 10^{17}$	$8{,}39 \cdot 10^{17}$	$5{,}39 \cdot 10^{19}$

Figura 6. Taula de càlculs del nombre de combinacions possibles segons el tipus de contrasenya.

(Elaboració pròpia)

2.4.1 Gràfiques segons la llargada de la contrasenya

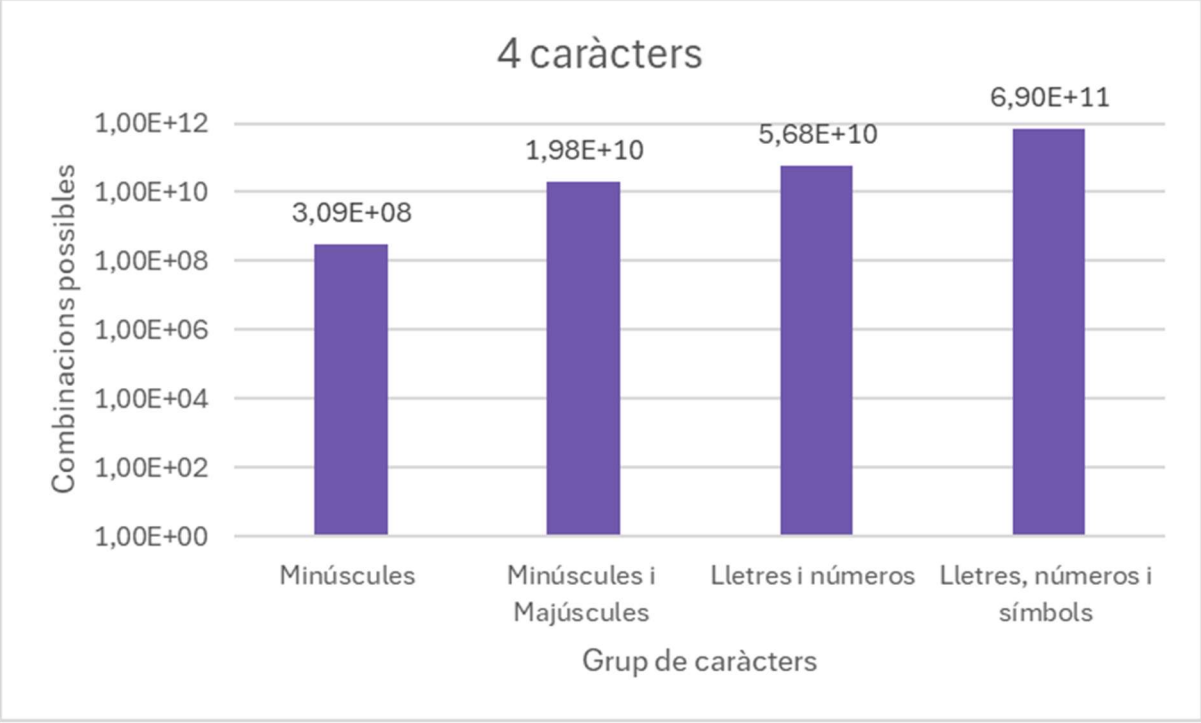

Figura 7. Gràfica del número de combinacions possibles en una contrasenya de 4 caràcters.

(Elaboració pròpia)

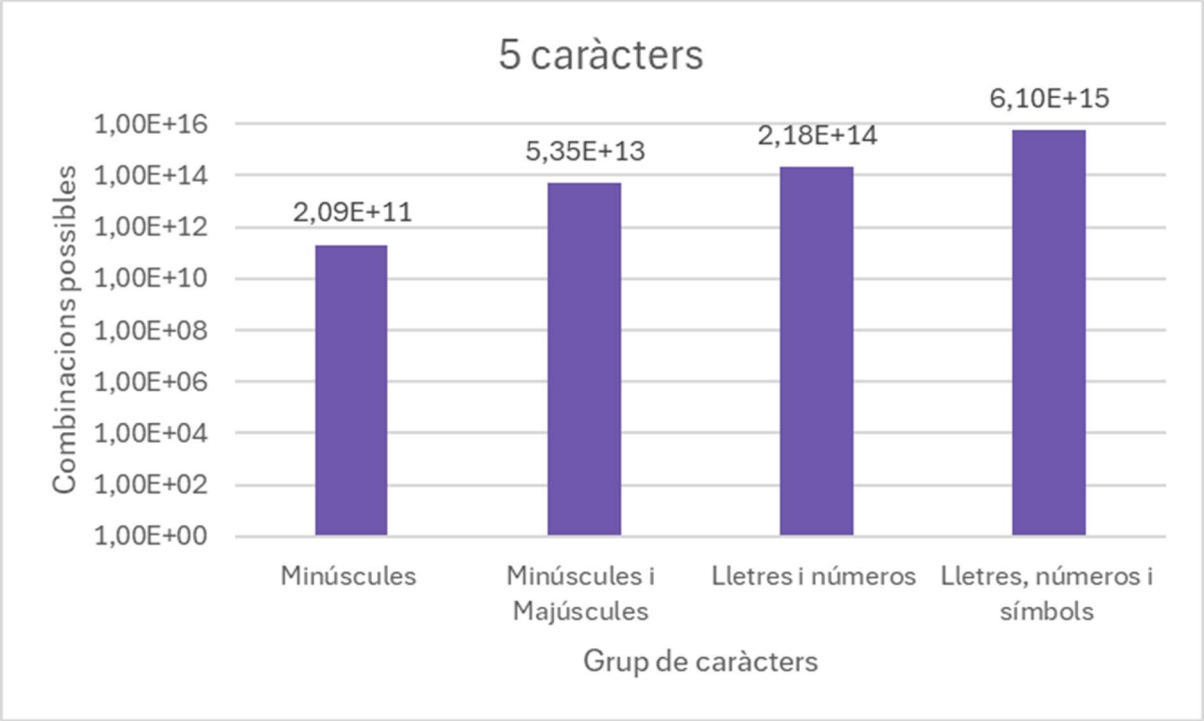

Figura 8. Gràfica del número de combinacions possibles en una contrasenya de 5 caràcters. (Elaboració pròpia)

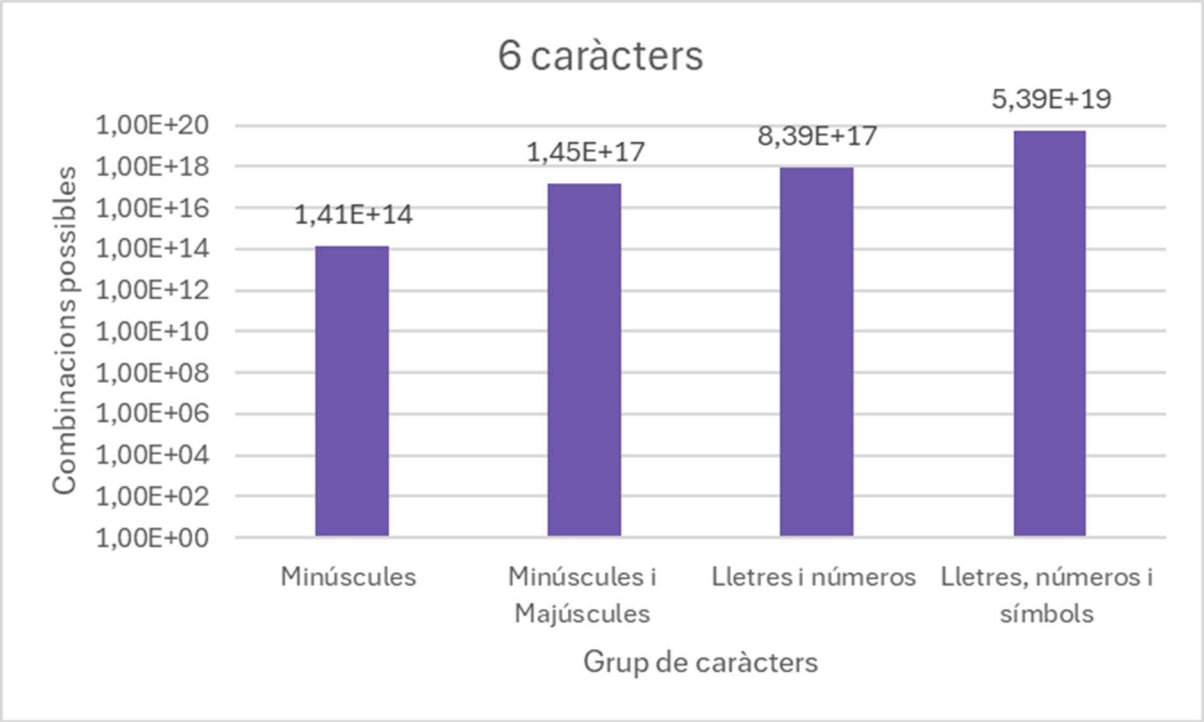

Figura 9. Gràfica del número de combinacions possibles en una contrasenya de 6 caràcters. (Elaboració pròpia)

En aquestes tres gràfiques es pot observar de manera més clara com l'increment de la complexitat del conjunt de caràcters deriva en un creixement exponencial del nombre de combinacions possibles. Cada gràfica permet comparar l'efecte de canviar l'amplitud del conjunt de caràcters, demostrat com petites modificacions en aquests paràmetres poden donar un augment significatiu en la dificultat de desxifrat. Cal destacar que el salt de només lletres minúscules i majúscules a l'ús de lletres i números és el més petit de tots, ja que el conjunt de caràcters de números és el més reduït.

2.4.2 Gràfiques segons el conjunt de caràcters

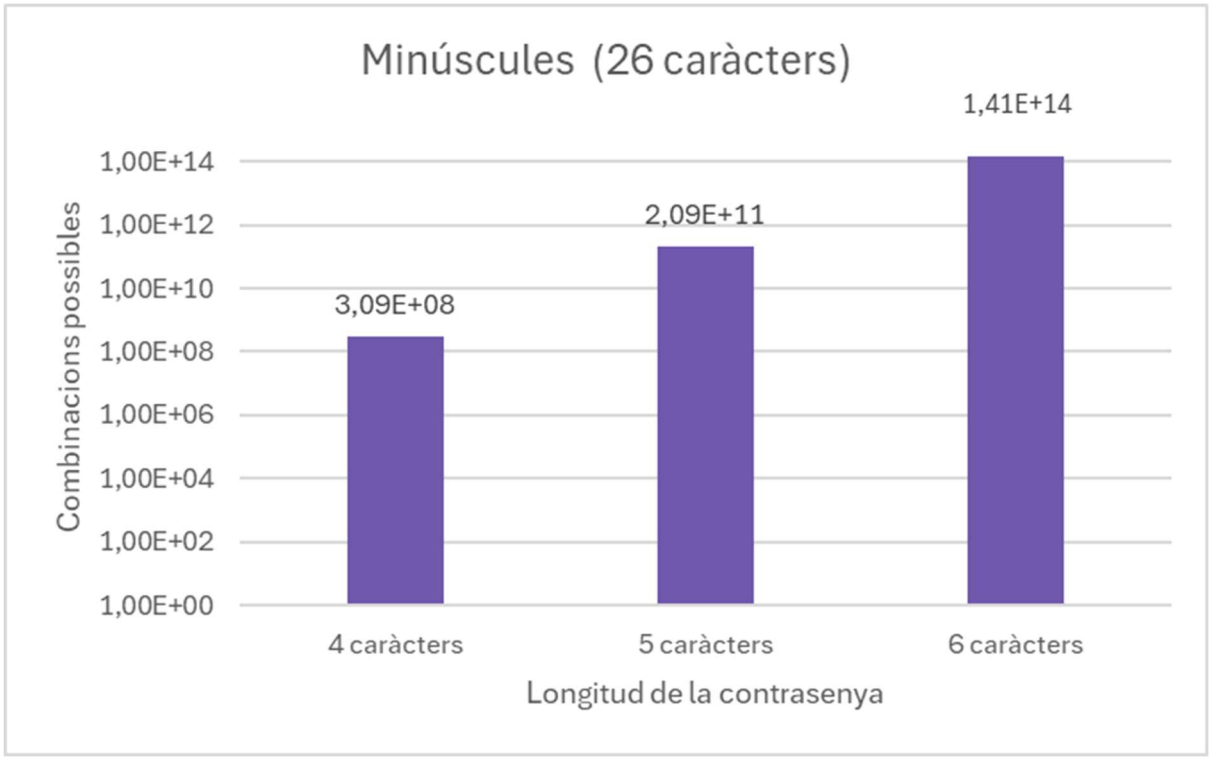

Figura 10. Gràfica del número de combinacions possibles utilitzant minúscules. (Elaboració pròpia)

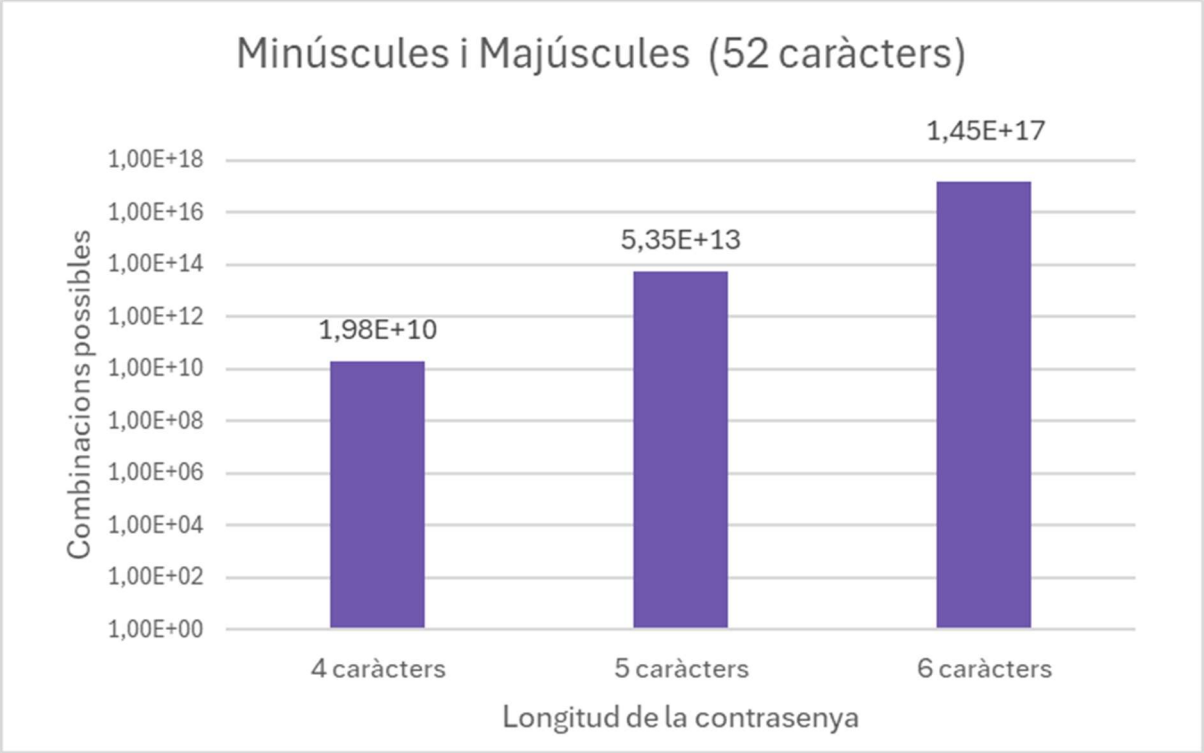

Figura 11. Gràfica del número de combinacions possibles utilitzant minúscules i majúscules. (Elaboració pròpia)

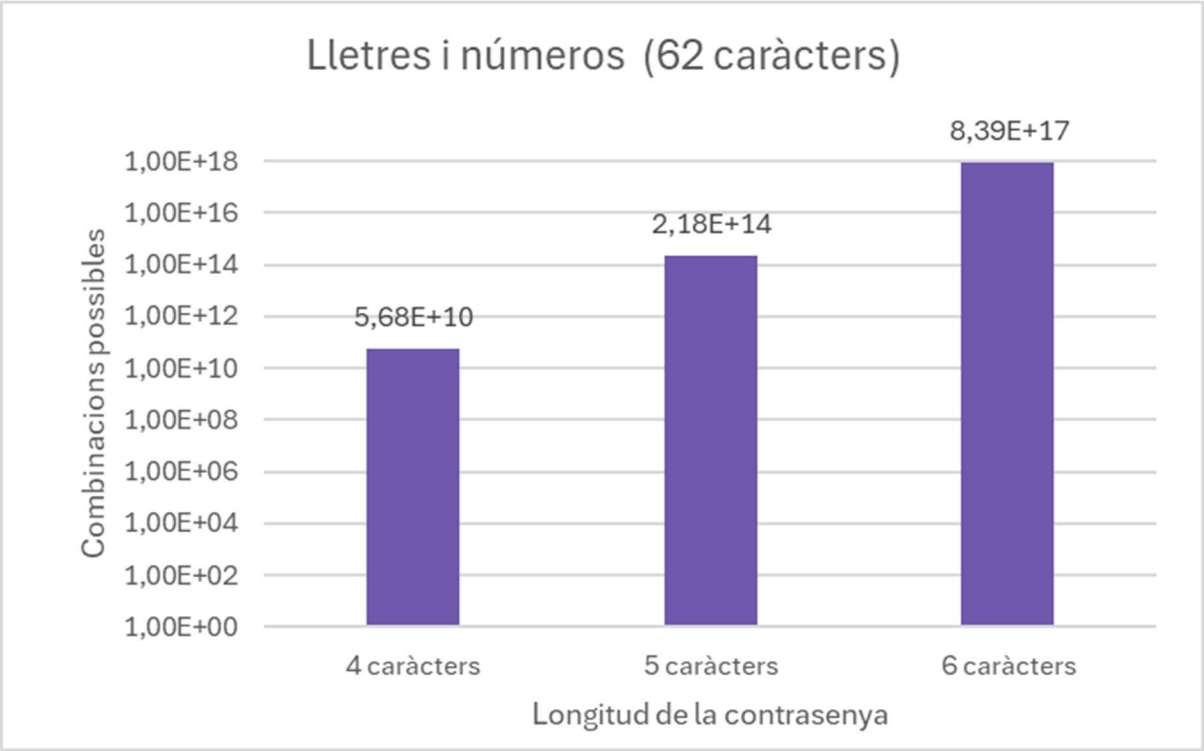

Figura 12. Gràfica del número de combinacions possibles utilitzant lletres i números. (Elaboració pròpia)

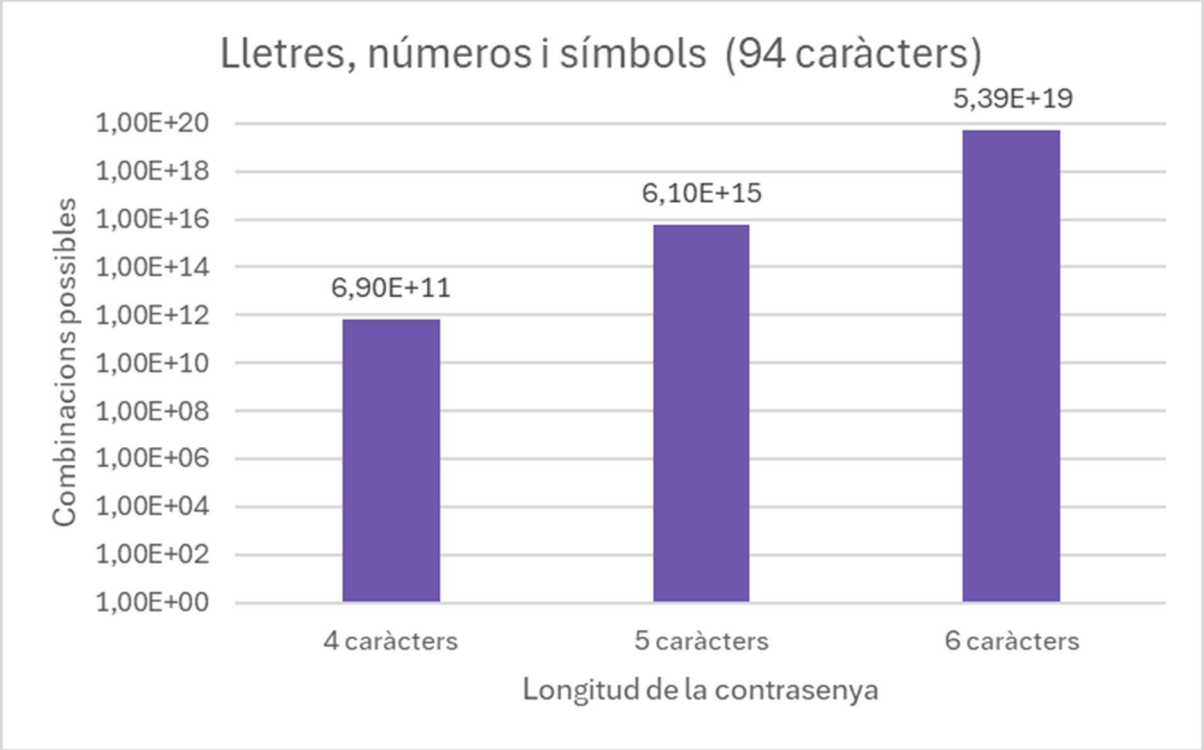

Figura 13. Gràfica del número de combinacions possibles utilitzant lletres, números i símbols. (Elaboració pròpia)

En aquestes quatre gràfiques es pot apreciar clarament com el nombre de combinacions possibles creix de manera exponencial a mesura que augmenta la longitud de la contrasenya. Cada gràfica mostra diferents conjunts de caràcters, evidenciant que, independentment de la complexitat del grup, fins i tot un increment d'una sol caràcter en la longitud provoca un augment molt significatiu de la dificultat de desxifrat. Això mostra la importància de la longitud com a factor clau.

2.5 Conclusions de l'anàlisi

D'aquest anàlisi se'n poden treure diverses conclusions sobre la seguretat de les contrasenyes i la relació entre el nombre de combinacions possibles, la llargada de la contrasenya i el conjunt de caràcters utilitzat. En primer lloc, cal destacar que el creixement és exponencial, no lineal: cada caràcter afegit augmenta les combinacions de manera molt substancial. Això implica que fins i tot petits increments en la llargada tenen un impacte molt significatiu en la seguretat.

A més, la quantitat de caràcters utilitzats és molt important. Utilitzar només minúscules és molt menys segur que utilitzar també majúscules, números i símbols. Com passa amb la longitud, una contrasenya amb més varietat de grups de caràcters és molt més forta. En aquest sentit, l'impacte dels símbols especials és especialment destacable. El salt en el nombre de combinacions quan s'hi afegeixen símbols és molt superior al que es dona en afegir només majúscules o números, ja que s'incorporen 32 caràcters nous al conjunt disponible.

Tot i això, l'anàlisi també mostra que les contrasenyes curtes són insegures, fins i tot si fan ús de molts tipus de caràcters. Una contrasenya de només 6 caràcters amb símbols pot semblar complexa, però continua sent realista poder trencar-la per força bruta amb una màquina potent.

En conjunt, aquestes conclusions demostren que s'ha de promoure l'ús habitual de contrasenyes d'almenys 10 caràcters i amb una combinació de majúscules, minúscules, números i símbols, especialment en contextos on la seguretat és fonamental.

3. Aplicació Experimental

Abans de començar cal recordar l'objectiu del treball per tal d'entendre millor aquest apartat. Aquest estudi busca comparar l'aplicació de la força bruta utilitzant 2 ordinadors, però tres contexts diferents: Força bruta sense paral·lelitzar, força bruta amb una paral·lelització de pocs *cores* i força bruta amb una paral·lelització de molts *cores*. Això amb l'objectiu de quantificar la influència de la supercomputació dins de l'àmbit de la seguretat informàtica.

3.1 Metodología

Per a la part pràctica d'aquest treball s'han realitzat una sèrie de proves per tal d'avaluar l'eficiència de diferents codis de força bruta sobre arxius ZIP xifrats. Com a unitat de "velocitat de prova" utilitzem les combinacions provades per segon (comb/s)

3.1.1 Preparació inicial

Abans de començar les proves, es va dur a terme una preparació per assegurar que els resultats fossin fiables i replicables. Primer, es van generar i organitzar els arxius ZIP corresponents a cada contrasenya a provar. Per reduir el possible biaix associat a la posició de la contrasenya dins del conjunt, es van crear tres arxius per cada cas experimental. Els noms dels arxius també es van pensar abans de crear-los per tal de facilitar l'organització: la nomenclatura consistia en la paraula *kobe* seguida de la longitud de la contrasenya (4, 5 o 6), el conjunt de caràcters (min, min+may…) i el nombre de prova. Per exemple: kobe5min+may2.zip. Finalment, a l'hora de generar els comprimits es va utilitzar sempre el mateix fitxer base per a tots els arxius ZIP, amb l'objectiu de minimitzar possibles variacions en el procés d'extracció i garantir que l'única variable rellevant fos la contrasenya a desxifrar. Seguidament, es van col·locar ventiladors addicionals per mantenir una temperatura estable durant l'execució, es va assegurar que l'únic programa en funcionament fos el codi de prova i es van configurar els nuclis del processador en mode *performance* per a la consistència del rendiment. Aquestes mesures van permetre establir un entorn controlat, minimitzant variables externes que podrien afectar a la velocitat de prova o el temps d'execució.

Eines i entorn experimental

Maquinari utilitzat

Per a la part experimental d'aquest treball s'han utilitzat dos ordinadors diferents:

1. MSI Cyborg 15 a13ve (ordinador portàtil personal):

   Processador: Intel Core i7-13620H, amb 10 nuclis (6 de tipus Performance i 4 de tipus Efficiency).

   Sistema operatiu: Fedora Linux 40

   Ús: Programar els codis i executar les proves amb i sense paral·lelització.

Figura 14, Ordinador MSI Cyborg 15 a13ve (msi.com, 205)

2. Marenostrum 5 (Subdivisió de 80 fils lògics, 75 utilitzats)

   Processadors: 2x Intel Sapphire Rapids 8480+

   Sistema operatiu: Linux basat en RHEL

   Ús: Executar el codi de força bruta amb paral·lelització per comparar el rendiment respecte l'ordinador personal.

Figura 15, Superordinador Marenostrum 5 (bsc.es, 2025)

Programari utilitzat

El software utilitzat a les proves consta de tres codis: un per a la força bruta sense paral·lelització, un per a la força bruta amb paral·lelització, i un altre per calcular el nombre de combinacions necessàries per desxifrar una contrasenya concreta, segons la seva longitud i el conjunt de caràcters que utilitza. Els codis estan escrits en Python perquè és un llenguatge senzill i llegible, amb llibreries que fan fàcil generar combinacions, manipular fitxers ZIP i paral·lelitzar. Per veure els codis vegeu els annexos 1, 2 i 3.

a) Codi comptador de combinacions

Aquest codi es el responsable de comptar el nombre de combinacions que son necessàries de provar fins a arribar a una contrasenya en específic

· Llibreries utilitzades (Línia 1)

string: Utilitzada per definir els diferents conjunts de caràcters possibles (minúscules, majúscules, números i símbols).

· Definició dels grups de caràcters (Línies 3 a 6)

En aquest bloc definim els caràcters que formen els quatre grups (minúscules, majúscules, números i símbols) i l'ordre en què s'utilitzen.

· Configuració del programa (Línies 8 a 12)

Es proporcionen al programa les variables amb les quals ha de treballar: el conjunt de caràcters, la contrasenya a analitzar i la longitud d'aquesta

· Funció *comb_to_password* (Línies 14 a 26)

Aquesta funció calcula la posició de la contrasenya dins de totes les combinacions possibles generades a partir del conjunt de caràcters donat. Matemàticament, cada contrasenya es pot interpretar com un nombre en base N, on N és la mida del conjunt de caràcters (len(charset)). Si considerem la contrasenya com una successió de caràcters c_0, c_1, c_{long-1}, la fórmula és:

$$posició = \sum_{i=0}^{long-} index(c_i) \cdot N^{long-i-1} + 1$$

On:

a) index(c_i) és la posició del caràcter c_i dins del conjunt *charset*.
b) $N^{long-i-1}$ és la potència de la base corresponent a la posició del caràcter dins la contrasenya (el caràcter més a l'esquerra té exponent més alt).
c) S'afegeix 1 al final perquè el comptador de combinacions comenci en 1 i no en 0.

b) Codi de força bruta sense paral·lelització:

Aquest codi es el responsable de desxifrar les contrasenyes dels arxius ZIP utilitzant el mètode de força bruta però sense paral·lelitzar el programa.

· Llibreries utilitzades (Línies 1 a 4)

zipfiles: Utilitzada per gestionar i extreure els arxius *.zip*.
itertools: Utilitzada per generar les combinacions possibles segons les variables estipulades.
string: Utilitzada per definir els diferents conjunts de caràcters possibles.
datetime: Utilitzada per mesurar el temps d'execució

·Definició dels grups de caràcters (Línies 6 a 9)

En aquest bloc es defineixen els caràcters que formen els quatre grups (minúscules, majúscules, números i símbols) i l'ordre en què s'utilitzen.

· Configuració del programa (Línies 11 a 16)

Es proporcionen al programa les variables amb les que ha de treballar: La ruta de l'arxiu a desxifrar, la longitud de la contrasenya i els grups de caràcters possibles.

· Funció *try_password* (Línies 18 a 23)

Aquesta funció és la responsable d'intentar extreure el fitxer xifrat amb una contrasenya determinada. Retorna *True* si l'extracció és correcta, i *False* en cas contrari.

· Funció principal *brute_force_zip* (Línies 25 a 44)

Genera totes les combinacions possibles amb itertools.product. Manté un comptador de les combinacions provades i atura l'execució quan es troba la contrasenya correcta. S'ha afegit un xivato que imprimeix cada X combinacions provades, mostrant el nombre de contrasenyes provades i la velocitat de prova (combinacions per segon). Quan es troba la contrasenya, es mostra per pantalla la contrasenya correcta, el nombre de combinacions provades i el temps d'execució amb decimals separats per coma.

· Mesura del temps d'execució (Línies 51 a 54)

S'utilitza datetime.now() per saber el moment d'inici i de finalització, i a partir d'això es calcula el temps total en segons, canviant el punt per una coma per facilitar l'ús de la dada a l'excel.

c) Codi de força bruta amb paral·lelització

Aquest codi es el responsable de desxifrar les contrasenyes dels arxius ZIP utilitzant el mètode de força bruta paral·lelitzant el programa.

· Llibreries utilitzades (Línies 1 a 7)

zipfile: Utilitzada per gestionar i extreure els arxius .zip.

itertools: Utilitzada per generar les combinacions possibles segons les variables estipulades.

string: Utilitzada per definir els diferents conjunts de caràcters possibles.

datetime: Utilitzada per mesurar el temps d'execució.

multiprocessing: Utilitzada per executar múltiples processos en paral·lel.

os i *sys*: Utilitzades per gestionar els processos i finalitzar l'execució quan es troba la contrasenya.

· Definició dels grups de caràcters (Línies 9 a 12)

En aquest bloc definim els caràcters que formen els quatre grups (minúscules, majúscules, números i símbols) i l'ordre en què s'utilitzen.

· Configuració del programa (Línies 14 a 21)

Es proporcionen al programa les variables amb les que ha de treballar: la ruta de l'arxiu a desxifrar, la longitud de la contrasenya, el grup de caràcters possibles, el nombre de processos a utilitzar i cada quan s'informa del progrés.

· Funció *try_password* (Línies 23 a 29)

Aquesta funció és la responsable d'intentar extreure el fitxer xifrat amb una contrasenya determinada. Retorna *True* si la contrasenya es correcta i *False* en cas contrari.

· Funció *worker* (Línies *31 a 56)*

Cada *worker* executa aquesta funció amb un prefix assignat. Genera totes les combinacions possibles i porta un comptador de contrasenyes provades. Imprimeix el nombre de contrasenyes provades i l'última contrasenya provada cada x combinacions (xivato). Si es troba la contrasenya correcta, aquesta es retorna, aturant el procés que l'ha trobat.

· Funció principal *main* (Línies 58 a 78)

S'inicialitza el temps d'inici i es crea un pool (un grup) de processos segons el nombre configurat abans. Cada prefix de dues lletres és un paquet que s'assigna a un worker dins del pool. El programa espera que alguna tasca retorni la contrasenya correcta. Quan això succeeix, s'imprimeix la contrasenya trobada, el temps total d'execució en segons (amb decimals separats per coma) i es finalitza el programa. Si cap worker troba la contrasenya, es tanca el pool i s'indica que no s'ha trobat.

Apart d'aquests tres codis, també s'ha utilitzat el programa HTOP per monitorizar la temperatura i rendiment a l'hora de realitzar les proves.

3.1.2 Optimització de la paral·lelització

Per maximitzar l'eficiència del càlcul de força bruta en el codi de força bruta paral·lelitzada es van implementar estratègies d'optimització de la paral·lelització.

Les tasques es van executar de manera asíncrona, permetent que cada worker processés el seu paquet de combinacions independentment. Aquesta estratègia evita que un worker hagi d'esperar que els altres acabin per agafar una altra tasca, minimitzant temps d'inactivitat i aprofitant millor els recursos.

Els paquets de treball es van generar utilitzant prefixes de dues lletres (per exemple, aa, ab, ac), de manera que cada worker generés totes les combinacions possibles que comencen per aquest prefix. Això fa que la distribució dels processos sigui més equitativa.

Com que les tasques són asíncrones i el problema está dividit en molts paquets, existeix el risc de que un mateix paquet sigui assignat a més d'un worker (*task duplication*). Tot i això, aquest fenomen és assumible degut a la alta velocitat d'execució i no impacta significativament en el temps total d'execució ni en els resultats finals.

Com que les tasques estan assignades de manera asíncrona, quan un worker troba la contrasenya correcta hi ha un petit temps de "sincronització", on la resta de workers encara poden estar executant les seves combinacions abans de "saber" que la contrasenya ja s'ha trobat i aturar-se. Aquest retard depen de la complexitat de la contrasenya. A contrasenyes complexes en aquest context, aquest temps de sincronització estarà entre 3 i 5 minuts com a molt.

3.2 Execució dels codis i registre de les dades

3.2.1 Proves seqüencials

Per a les proves sense paral·lelització, es va llançar el codi al portàtil de manera seqüencial sobre cada arxiu ZIP, amb un límit màxim d'execució de tres hores per que no era necessari fer totes les proves, ja que a partir d'aquestes dades es podien estimar la resta. Durant aquestes execucions es van registrar el temps total i el nombre de combinacions provades fins a trobar la contrasenya. A partir de totes les execucions completes dins d'aquest límit i les combinacions per segon extretes del chivato a tots els casos, es van calcular les mitjanes de velocitat i, utilitzant el comptador de combinacions, es van estimar els temps necessaris per a desxifrar la resta de contrasenyes.

3.2.2 Proves paral·lelitzades

Les proves amb paral·lelització es divideixen en dos, les realitzades al portàtil i les realitzades al Marenostrum 5.

Al portàtil es va fer una paral·lelització de 6 processos, i es van realitzar proves en tots els casos de contrasenyes de longitud 4. Aquesta decisió va permetre obtenir dades suficientes per comparar amb les execucions seqüencials i extrapolar-les per a contrasenyes més llargues. Al Marenostrum 5, amb una paral·lelització de 75 processos, es van realitzar només dues proves a contrasenyes de longitud més gran, la que utilitza caràcters minúscules i majúscules de longitud 5, i la que utilitza minúscules de longitud 6.

3.2.3 Dificultats durant l'experimentació

En un primer moment totes les proves van ser executades al meu portàtil sense tenir en compte que els codis estaven utilitzant *Efficiency Cores*. Això va provocar que el rendiment fos molt més lent i que els resultats obtinguts no fossin els més adequats ni representatius per al treball.

Un cop solucionat el problema, vaig repetir totes les proves assignant l'execució a *Performance Cores*, que aprofiten del tot la potència de l'ordinador. D'aquesta manera, els resultats van ser més consistents i coherents amb la teoria i amb l'objectiu del treball.

3.2.4 Registre de dades i resultats

Totes les dades recollides dels experiments han estat plasmades a dos arxius d'excel: un per a les execucions seqüencials i un per a les paral·lelitzades. Per veure les gràfiques, vegeu els annexos 3 i 4.

A partir d'aquests resultats se'n poden extreure diverses observacions rellevants i significatives per al treball:

a) Velocitat estable independentment de la complexitat de la contrasenya:

S'esperava que, a mesura que la contrasenya fos més complexa, les combinacions per segon disminuirien. Però, s'ha observat que les combinacions per segon es mantenien pràcticament constants, independentment del conjunt de caràcters o la longitud.

b) Debilitat de les contrasenyes que comencen per "a":

En els atacs seqüencials, les contrasenyes que comencen per la lletra "a" (o per les primeres lletres de l'alfabet) són més febles, ja que els algoritmes proven les combinacions en ordre alfabètic. Això fa que, en màquines petites on s'executa de manera seqüencial, aquestes contrasenyes siguin significativament més fàcils de trencar. Encara que en superordinadors com el MareNostrum la paral·lelització distribueix el treball entre processos, la manera de definir els paquets de treball pot fer que aquesta vulnerabilitat encara existeixi. En el nostre cas, com que els paquets es defineixen amb prefixos de dues lletres començant per "aa, ab, ac...", les contrasenyes que comencen amb lletres properes al principi de l'alfabet continuen sent més febles.

c) Diferència de velocitat a les 3 màquines

En execució seqüencial al portàtil personal, la velocitat es d'aproximadament 8.000 comb/s. Quan es paral·lelitza el procés a 6 workers, la velocitat per cada worker disminueix fins a uns 4.500 comb/s, demostrant la despesa de recursos associat a la gestió dels processos paral·lels. Si comparem amb el Marenostrum 5, observem que cada procés individual realitza molt menys combinacions per segon, al voltant de 1.760 comb/s per worker.

Tot i això, la gran quantitat de processos (75) permet que el rendiment global sigui molt superior, demostrant que la paral·lelització massiva compensa la menor velocitat individual dels processos. Aquesta observació és important a l'hora d'analitzar l'eficiència i escalar les proves a contrasenyes més llargues.

3.3 ??? (coses que m'he trobat i no tenia plantejades)

Durant l'experiment s'han observat col·lisions de contrasenyes en fitxers ZIP xifrats amb ZIP Crypto. Amb col·lisió de contrasenyes es vol dir que per al mateix arxiu hi ha més contrasenyes correctes, a part de la que se li ha posat inicialment.
Aquest fenomen es produeix perquè el xifrat Zip Crypto utilitzat no genera claus internes úniques per a cada contrasenya: contrasenyes diferents poden derivar la mateixa clau interna.

ZIP Crypto utilitza una versió modificada de RC4 (un algorisme de xifrat, és a dir, un mètode per codificar dades). En aquest sistema, la contrasenya genera una clau inicial (key) que es combina amb un vector d'inicialització (IV) de 12 bytes, que és un petit conjunt de dades guardat a la capçalera (header) del fitxer ZIP. La capçalera és la part inicial del fitxer que conté informació sobre el fitxer, com el seu nom, la seva mida i com s'ha de desxifrar, abans de començar a llegir el contingut real. Gràcies a l'IV, cada xifrat és una mica diferent, encara que s'utilitzi la mateixa contrasenya. Tot i això, l'espai de claus intern és limitat (2^{32} combinacions), cosa que significa que, de vegades, contrasenyes diferents poden generar la mateixa clau RC4.

Durant les proves de força bruta es pot donar una contrasenya com a "correcta" encara que no sigui l'original, degut a aquesta coincidència de claus. Aquest descobriment evidencia les limitacions de ZIP Crypto, ja que la mateixa clau interna pot correspondre a diverses contrasenyes. Veient això, es pot veure la importància d'utilitzar mètodes de xifrat més forts, com AES-256 (de grau militar), per garantir la seguretat dels fitxers ZIP en entorns reals.

Un exemple on es pot observar aquest fenòmen és en el primer arxiu del tipus minúscules, majúscules i números de 4 caràcters. La contrasenya original per aquest arxiu era "aF9m", però al executar la força bruta, ens dóna com a resultat "aaaH" (que es desxifra molt abans que "aF9m"). Quan es comproven les contrasenyes, les dues donen com a correctes.

3.4 Conclusions

A partir de l'experimentació realitzada, es poden extreure diverses conclusions tant des del punt de vista tècnic com des de la perspectiva de ciberseguretat. En primer lloc, les proves han confirmat que la paral·lelització millora de manera molt significativa el rendiment de la força bruta. Tot i que la velocitat per *worker* disminueix, el rendiment total augmenta notablement amb l'augment dels *workers*. Això demostra que aquest tipus d'estratègia és escalable i permet atacar de manera més ràpida contrasenyes complexes quan es té accés als ordinadors amb capacitat de paral·lelització massiva.

Tot i això, és important tenir en compte que, en l'actualitat, l'accés a superordinadors com el Marenostrum està controlat i limitat a usuaris autoritzats, per això l'ús d'aquestes màquines amb motivacions malicioses no és una preocupació real actualment per a la seguretat de contrasenyes de l'usuari comú.

En segon lloc, els resultats de les execucions confirmen la importància tant de la longitud com de la varietat de caràcters utilitzats a les contrasenyes. Contrasenyes més llargues i que combinen majúscules, minúscules, números i símbols tenen una resistència molt més alta als atacs de força bruta, mentre que les contrasenyes curtes o basades en un conjunt reduït de caràcters són molt més vulnerables.

Finalment, les proves han evidenciat la importància de l'alfabet utilitzat per a les contrasenyes. Conjunts de caràcters més amplis i complexos, com els que es troben en idiomes amb alfabets més llargs o sistemes de caràcters com el japonès, incrementen exponencialment el nombre de combinacions possibles i fan que l'atac per força bruta sigui molt menys possible.

En resum, la paral·lelització pot accelerar considerablement l'atac de força bruta, però factors com la longitud, la complexitat i l'amplitud de l'alfabet de les contrasenyes continuen sent els elements més determinants per garantir la seguretat.

Com a línia d'ampliació al treball, es podria considerar aprofundir en l'estudi de la vulnerabilitat del xifratge Zip crypto per veure fins a on es podria explotar. A més, també es podria seguir investigant quina seria la millor optimització paral·lela per tal de maximitzar la força bruta.

4. Referències

Alkhwaja, I., Mohammed, A., Alkhwaja, A., Alghamdi, M., Hussam Abahussain, Faisal Alfawaz, Abdullah Almurayh, & Min-Allah, N. (s.d.). Password Cracking with Brute Force Algorithm and Dictionary Attack Using Parallel Programming. *2023*.

Bishop, M. (2018). *Computer Security: Art and Science* (2ª, Vol. 1-1). Addison-Wesley Professional.

BSC. (2025a). *Material Docent Bojos per la Supercomputació* [PowerPoint].

Cloudflare. (s.d.-a). *What is a Web Application Firewall*. https://www.cloudflare.com/learning/ddos/glossary/web-application-firewall-waf/

Hockney, R. (s.d.). *Performance parameters and benchmarking of supercomputers*.

LENOVO. (s.d.-b). *LENOVO Glossary* [Glossari]. LONOVO Glossary. https://www.lenovo.com/us/en/glossary/

Microsoft. (s.d.-c). *What is: Multifactor Authentication*. Microsoft Support. https://support.microsoft.com/en-us/topic/what-is-multifactor-authentication-e5e39437-121c-be60-d123-eda06bddf661?utm_source=chatgpt.com

Miller, M. (2024, de maig de). Password Cracking 101: Attacks & Defenses Explained [Blog tècnic / empresa]. *BeyondTrust Blog*. https://www.beyondtrust.com/blog/entry/password-cracking-101-attacks-defenses-explained

Mitre Att&ck. (2023, juny 6). *CAPEC—Common Attack Pattern Enumerations and Classifications* [Base de dades]. CAPEC - Common Attack Pattern Enumerations and Classifications. https://capec.mitre.org/

NICCS - CISA. (2025b, de maig de). *NICCS Glossary* [Glossari]. Glossary. https://niccs.cisa.gov/resources/glossary

Norton. (s.d.-d). *Gestor de Contraseñas*. https://es.norton.com/feature/password-manager

Stallings, W. (2018). *Effective Cybersecurity* (1ª, Vol. 1-1). Addison-Wesley Professional.

Tirado, E., Turpin, B., Beltz, C., Phillip Roshon, Judge, R., & Gagneja, K. (2018). *A New Distributed Brute-Force Password Cracking Technique*. 117-127. https://link.springer.com/chapter/10.1007/978-3-319-94421-0_9#citeas

UPC. (2013). *Presentacions Paral·lelisme UPC* [PowerPoint].

5. Annexos

Annex 1

Enllaç que porta al codi comptador de combinacions dins el repositori de GitHub del projecte

https://github.com/ablesphinx/Impact-of-Supercomputing-on-ZIP-Password-Cracking/blob/main/python%20codes/combination_counter.py

Annex 2

Enllaç que porta al codi de força bruta sense paral·lelització dins el repositori de GitHub del projecte

https://github.com/ablesphinx/Impact-of-Supercomputing-on-ZIP-Password-Cracking/blob/main/python%20codes/sequential_zip_bruteforce.py

Annex 3

Enllaç que porta al codi de força bruta amb paral·lelització dins el repositori de GitHub del projecte

https://github.com/ablesphinx/Impact-of-Supercomputing-on-ZIP-Password-Cracking/blob/main/python%20codes/parallelized_zip_bruteforce.py

Annex 3

Gràfica de resultats de les proves sense paral·lelització dins el repositori de GitHub del projecte

https://github.com/ablesphinx/Impact-of-Supercomputing-on-ZIP-Password-Cracking/blob/main/data_tables/sequential_results_catalan.xlsx

Annex 4

Gràfica de resultats de les proves amb paral·lelització dins el repositori de GitHub del projecte

https://github.com/ablesphinx/Impact-of-Supercomputing-on-ZIP-Password-Cracking/blob/main/data_tables/parallel_results_catalan.xlsx