



Ciencia Latina Revista Científica Multidisciplinar, Ciudad de México, México.
ISSN 2707-2207 / ISSN 2707-2215 (en línea), septiembre-octubre 2026,
Volumen 10, Número 5.

https://doi.org/10.37811/cl_rcm.v10i5

DEMOCRATIZANDO LA SEGURIDAD: LECCIONES PRÁCTICAS DE FUERZA BRUTA Y DICCIONARIO PARA MITIGAR RIESGOS EN PYMES

DEMOCRATIZING SECURITY: PRACTICAL LESSONS ON BRUTE-FORCE AND DICTIONARY ATTACKS FOR MITIGATING RISKS IN SMES

Marisela Palacios Reyes

Instituto Tecnológico de Ciudad Juárez, Mexico

Juan Manuel Bernal Ontiveros

Instituto Tecnológico de Ciudad Juárez, Mexico

Noé Ramón Rosales Morales

Instituto Tecnológico de Ciudad Juárez, Mexico

Verónica Farías Veloz

Instituto Tecnológico de Ciudad Juárez, Mexico

Montserrat Valdez Escobedo

Instituto Tecnológico de Ciudad Juárez, Mexico

Democratizando la Seguridad: Lecciones Prácticas de Fuerza Bruta y Diccionario para Mitigar Riesgos en PyMES

Marisela Palacios Reyes¹

marisela.pr@cdjuarez.tecnm.mx
<https://orcid.org/0000-0003-2830-5829>

Tecnológico Nacional de México
Instituto Tecnológico de Ciudad Juárez
Mexico

Juan Manuel Bernal Ontiveros

juan.bo@cdjuarez.tecnm.mx
<https://orcid.org/0000-0002-3819-5750>

Tecnológico Nacional de México
Instituto Tecnológico de Ciudad Juárez
Mexico

Noé Ramón Rosales Morales

Noe.rm@cdjuarez.tecnm.mx
<https://orcid.org/0000-0003-4526-3448>

Tecnológico Nacional de México
Instituto Tecnológico de Ciudad Juárez
Mexico

Verónica Farías Veloz

veronica.fv@cdjuarez.tecnm.mx
<https://orcid.org/0000-0001-8147-1573>

Tecnológico Nacional de México
Instituto Tecnológico de Ciudad Juárez
Mexico

Montserrat Valdez Escobedo

L23111475@cdjuarez.tecnm.mx
<https://orcid.org/0009-0002-1746-0284>

Tecnológico Nacional de México
Instituto Tecnológico de Ciudad Juárez
Mexico

RESUMEN

Las Pequeñas y Medianas Empresas (PyMES) enfrentan crecientes amenazas a su seguridad perimetral debido a la falta de políticas estrictas en la gestión de credenciales y a la reutilización de datos de autenticación predecibles. Este artículo presenta un análisis experimental y educativo sobre cómo la debilidad en las claves de acceso compromete la confidencialidad organizacional a través de tres técnicas criptanalíticas fundamentales: ataques de fuerza bruta, ataques de diccionario y ataques de texto plano conocido (KPA). Mediante un entorno de laboratorio controlado en plataformas Linux (AlmaLinux y Kali Linux) e ingeniería de software especializada (OpenSSL, John the Ripper y CrypTool 2), se evaluó el impacto del tamaño del espacio de claves, el uso de wordlists contextuales y la correlación entre texto claro y texto cifrado bajo el esquema AES-256-CBC. Los resultados demuestran la viabilidad técnica de descifrar esquemas basados en la predictibilidad humana e ilustran cómo el conocimiento de patrones estandarizados permite la reconstrucción de claves mediante análisis matemático y lógico. Finalmente, se formulan recomendaciones prácticas defensivas adaptadas a entornos de recursos limitados, enfatizando la implementación de algoritmos de derivación lentos (PBKDF2, Argon2), el uso de salts y políticas de entropía elevada (passphrases) para transferir este conocimiento técnico en un beneficio directo para la protección comunitaria e industrial.

Palabras clave: criptografía aplicada, seguridad perimetral, fuerza bruta, ataque por diccionario, Texto conocido (KPA)

¹ Autor principal

Correspondencia: marisela.pr@cdjuarez.tecnm.mx

Democratizing Security: Practical Lessons on Brute-Force and Dictionary Attacks for Mitigating Risks in SMEs

ABSTRACT

Small and Medium-sized Enterprises (SMEs) face growing threats to their perimeter security due to a lack of strict credential management policies and the reuse of predictable authentication data. This article presents an experimental and educational analysis of how weak access credentials compromise organizational confidentiality through three fundamental cryptanalytic techniques: brute-force attacks, dictionary attacks, and known-plaintext attacks (KPA). Using a controlled laboratory environment—based on Linux platforms (AlmaLinux and Kali Linux) and specialized software (OpenSSL, John the Ripper, and CrypTool 2)—the study evaluates the impact of key-space size, the use of contextual wordlists, and the correlation between plaintext and ciphertext under the AES-256-CBC scheme. The results demonstrate the technical feasibility of decrypting schemes that rely on human predictability and illustrate how knowledge of standardized patterns enables the reconstruction of keys through mathematical and logical analysis. Finally, the article offers practical defensive recommendations tailored for resource-constrained environments, emphasizing the implementation of slow key-derivation functions (such as PBKDF2 and Argon2), the use of salts, and policies requiring high-entropy credentials (passphrases), thereby translating this technical insight into direct benefits for community and industrial protection.

Keywords: applied cryptography, perimeter security, brute force, dictionary attack, known-plaintext attack (KPA)

Artículo recibido 20 agosto 2026

Aceptado para publicación: 20 setiembre 2026



INTRODUCCIÓN

En la era de la transformación digital, las Pequeñas y Medianas Empresas (PyMES) representan un pilar económico fundamental; sin embargo, continúan siendo el blanco principal de ciberataques debido a limitaciones en presupuesto, infraestructura y capacitación en ciberseguridad (National Institute of Standards and Technology [NIST], 2020). La seguridad perimetral de una organización depende en gran medida de la solidez de sus esquemas de autenticación y cifrado. No obstante, la persistencia de patrones predecibles en la elección de contraseñas y la reutilización de credenciales por parte de los usuarios representan el eslabón más vulnerable de la cadena de protección (Stallings, 2017).

Los ataques criptanalíticos de fuerza bruta y de diccionario explotan precisamente esta predictibilidad humana y el poder de cómputo moderno para derivar o revertir claves de acceso (Solar Designer, 2023; Steube, 2022). Mientras que la fuerza bruta efectúa una búsqueda exhaustiva probando todas las combinaciones posibles dentro de un espacio de clave finito (Stallings, 2017), los ataques de diccionario optimizan este proceso priorizando términos comunes, combinaciones frecuentes y listas de datos previamente filtrados (wordlists) (Biryukov et al., 2016; Steube, 2022). Por otro lado, los ataques de texto plano conocido (Known-Plaintext Attacks, KPA) permiten a un atacante que posee pares de datos en claro y cifrados deducir la clave o descubrir debilidades estructurales en el algoritmo utilizado (Gate Learn, 2023; Stallings, 2017).

A pesar del desarrollo de estándares avanzados como el Estándar de Cifrado Avanzado (AES) y funciones de derivación de claves resistentes (PBKDF2, Argon2), muchas implementaciones operativas en PyMES sufren de configuraciones obsoletas o contraseñas de baja entropía que facilitan el compromiso de datos confidenciales (Biryukov et al., 2016; NIST, 2020). Este artículo aborda dicha problemática desde una perspectiva práctico-educativa. A través de pruebas de laboratorio en entornos controlados de código abierto (AlmaLinux, Kali Linux, OpenSSL, John the Ripper y CrypTool 2), se analiza cómo las vulnerabilidades en la longitud y complejidad de las claves impactan la resistencia criptográfica, proporcionando recomendaciones defensivas aplicables que contribuyan a la resiliencia tecnológica comunitaria e industrial.

METODOLOGÍA

La presente investigación adoptó un enfoque empírico-experimental de carácter cuantitativo, estructurado en un entorno de laboratorio virtual controlado para auditar la resistencia de sistemas criptográficos ante ataques basados en credenciales débiles y fallas de implementación (Stallings, 2017). Con el fin de evaluar la brecha entre la teoría criptográfica formal y la práctica operativa en organizaciones con recursos limitados, la experimentación se dividió en tres fases metodológicas: (1) análisis de cifrado simétrico y entropía de claves mediante OpenSSL; (2) evaluación del impacto temporal del criptoanálisis por fuerza bruta en contenedores protegidos; y (3) modelado de ataques de diccionario utilizando heurísticas de comportamiento humano (Biryukov et al., 2016; O'Flynn & Chen, 2014).

Arquitectura del Entorno Experimental

Para garantizar la validez interna de las pruebas y la reproducibilidad de los resultados, se configuró una arquitectura de red dual basada en hosts virtuales aislados:

Estación de Cifrado y Análisis Local (AlmaLinux OS): Entorno destinado a la simulación de políticas de almacenamiento seguro, pruebas de derivación de claves y análisis de bloques binarios (National Institute of Standards and Technology [NIST], 2020).

Estación de Criptoanálisis y Auditoría (Kali Linux): Plataforma equipada con herramientas especializadas en pruebas de penetración y recuperación de hashes (Steube, 2022; Solar Designer, 2023).

Herramientas y Frameworks Criptográficos

OpenSSL: Utilizado para la ejecución de algoritmos de cifrado simétrico robustos bajo el estándar AES en modo CBC, implementando funciones de expansión de claves basadas en Password-Based Key Derivation Function 2 (PBKDF2) (Stallings, 2017).

John the Ripper (v1.9.0-jumbo-1): Framework de código abierto optimizado para el agrietamiento de hashes mediante procesamiento paralelo (OpenMP), empleado para evaluar los tiempos de respuesta ante claves de diferentes longitudes (Solar Designer, 2023).

CrypTool 2: Entorno educativo de código abierto enfocado en la visualización de algoritmos criptográficos clásicos y modernos, empleado en la fase de modelado conceptual del criptoanálisis (Gate Learn, 2023).

Procedimientos Experimentales

Fase 1: Evaluación del Cifrado Simétrico y Estructura de Bloques

Se evaluó el comportamiento de la confidencialidad ante la entropía de las claves de acceso mediante el cifrado de un archivo de texto plano de prueba (mensaje.txt con la directiva LOGIN ADMIN) (NIST, 2020).

1 Generación de credenciales de contraste: Se contrastaron perfiles de contraseñas de baja entropía (password123) frente a cadenas de alta entropía (H7r9Lq52VpXm), validando su integridad mediante el cálculo de huellas hash SHA-256 (Biryukov et al., 2016).

```
alumna@almalinux-lab:~/Lab_KPAS$ cat debil.txt
password123
alumna@almalinux-lab:~/Lab_KPAS$ cat robusta.txt
H7r9Lq52VpXm
alumna@almalinux-lab:~/Lab_KPAS$ sha256sum debil.txt
57db1253b68b6802b59a969f750fa32b60cb5cc8a3cb19b87dac28f541dc4e2a  debil.txt
alumna@almalinux-lab:~/Lab_KPAS$ sha256sum robusta.txt
6593e0a5453e836616e028aba7c4cf047b1149e0368337ef58c9a669bb78b574  robusta.txt
alumna@almalinux-lab:~/Lab_KPAS$ cat mensaje.txt
LOGIN ADMIN
```

2 Cifrado y advertencias de obsolescencia: Se ejecutó el cifrado simétrico utilizando AES-256-CBC, registrando las alertas del sistema ante la ausencia de parámetros de derivación robustos:

```
openssl enc -aes-256-cbc -salt -pbkdf2 -in mensaje.txt -out mensaje_seguro.enc -k password123
```

3 Inspección de bajo nivel en formato hexadecimal (xxd): Mediante la herramienta de inspección binaria, se comprobó la estructura interna del archivo cifrado para identificar el encabezado de aleatorización (Salt), el cual previene ataques basados en tablas precalculadas (Stallings, 2017):

```
alumna@almalinux-lab:~/Lab_KPAS$ xxd mensaje.enc
00000000: 5361 6c74 6564 5f5f 3ff8 a5b3 2cf4 08c9  Salted_?...,...
00000010: ff64 6663 fff0 4770 8cf1 c8df 1888 9197  .dfc..Gp.....
alumna@almalinux-lab:~/Lab_KPAS$ xxd mensaje_seguro.enc
00000000: 5361 6c74 6564 5f5f f111 f450 e0f3 016e  Salted_...P...n
00000010: 89c1 838d 352d 3319 6b2e f705 19f3 eb51  ....5-3.k.....Q
```

```

alumno@almalinux-lab:~/Lab_KPA$ openssl enc -d -aes-256-cbc -pbkdf2 -in mensaje_seguro.enc -out mensaje_recuperado.txt -k
H7r9Lq52VpXm
Invalid command 'enc-d'; type 'help' for a list.
alumno@almalinux-lab:~/Lab_KPA$ openssl enc -d -aes-256-cbc -pbkdf2 -in mensaje_seguro.enc -out mensaje_recuperado.txt -
k H7r9Lq52VpXm
alumno@almalinux-lab:~/Lab_KPA$ cat mensaje_recuperado.txt
LOGIN_ADMIN

```

Fase 2: Medición de Rendimiento en Ataques de Fuerza Bruta

Siguiendo los principios metodológicos de evaluación de espacios de claves (Stallings, 2017), se estructuraron dos escenarios experimentales sobre contenedores comprimidos protegidos:

Escenario A: Contenedor con clave corta de 5 caracteres numéricos (12345).

```

(kali㉿kali)-[~/Desktop/proteccion]
└─$ zip2john archivoNprotegido.zip > hash.txt
ver 1.0 efh 5455 efh 7875 archivoNprotegido.zip/mensaje3.txt PKZIP Encr: 2b c
hk, TS_chk, cmplen=36, decmplen=24, crc=262E111F ts=93B6 cs=93b6 type=0

(kali㉿kali)-[~/Desktop/proteccion]
└─$ cat hash.txt
archivoNprotegido.zip/mensaje3.txt:$pkzip$1*2*2*0*24*18*262e111f*0*46*0*24*93
b6*daf563e216d56a95c5e7ca9fa83f220a15c3c8fd95e158dbb068701bf891977122a68249*$
/pkzip$:mensaje3.txt:archivoNprotegido.zip::archivoNprotegido.zip

(kali㉿kali)-[~/Desktop/proteccion]
└─$ john hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
12345      (archivoNprotegido.zip/mensaje3.txt)
1g 0:00:00:00 DONE 2/3 (2026-09-24 18:32) 7.142g/s 353271p/s 353271c/s 353271
C/s 123456..ferrises
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

```

Escenario B: Contenedor con clave extendida de 9 caracteres (123451234).

```
(kali㉿kali)-[~/Desktop/proteccionA]
$ cat hash.txt
archivoP.zip/mensaje4.txt:$pkzip$1*2*2*0*2e*22*9b28e38*0*46*0*2e*99fa*329a889
3ddcef58a54d5ece609b5df4d229b64b29d226893528a9c54714320440b5aee7544e1f5ecfc17
a889cc72*$/pkzip$:mensaje4.txt:archivoP.zip::archivoP.zip

(kali㉿kali)-[~/Desktop/proteccionA]
$ john hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
Proceeding with incremental:ASCII
123451234 (archivoP.zip/mensaje4.txt)
1g 0:00:01:02 DONE 3/3 (2026-09-24 19:18) 0.01600g/s 8318Kp/s 8318Kc/s 8318KC
/s gsixmjc..123451433
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Tras la extracción de los hashes con la utilidad zip2john, se ejecutó el proceso de iteración masiva con john, registrando métricas de rendimiento en velocidad de procesamiento y tiempo transcurrido (Solar Designer, 2023)

Fase 3: Modelado del Ataque de Diccionario y Heurísticas Humanas

El criptoanálisis basado en diccionario se formalizó mediante la comparación iterativa entre un conjunto candidato de palabras \$w_i\$ pertenecientes a una lista predefinida \$D\$ (como repositorios históricos tipo RockYou) y el hash objetivo \$H(p)\$ (Steube, 2022). Se evaluó la eficiencia de las reglas de transformación gramatical (mutaciones leet, adición de sufijos y patrones de fechas), demostrando que la predictibilidad humana reduce exponencialmente el espacio de búsqueda en comparación con la fuerza bruta exhaustiva (Biryukov et al., 2016; Gate Learn, 2023).

Aplicar_heuristicas(palabra): Simula el motor de reglas de herramientas profesionales (como Hashcat o John the Ripper), demostrando cómo una sola palabra común (ej. password) se multiplica exponencialmente en decenas de variantes lógicas que los humanos suelen elegir por facilidad de memorización.

Ataque_diccionario_heuristico(...): Representa la validación iterativa comparando el hash de cada candidato generado contra el hash objetivo (hash_objetivo), evidenciando por qué la predictibilidad humana reduce drásticamente el espacio de búsqueda en comparación con la fuerza bruta pura.

```

Python
import hashlib

def aplicar_heurísticas(palabra):
    """
    Aplica reglas de mutación basadas en el comportamiento humano predecible
    (Capitalización, Leet Speak y adición de sufijos comunes).
    """
    candidatos = {palabra}

    # 1. Variaciones de mayúsculas/minúsculas
    candidatos.add(palabra.capitalize())
    candidatos.add(palabra.upper())

    # 2. Sustituciones por Leet Speak básico (ej. a -> @, e -> 3, s -> $)
    reemplazos = {'a': '@', 'e': '3', 'i': '1', 'o': '0', 's': '$'}
    palabra_leet = palabra
    for original, reemplazo in reemplazos.items():
        palabra_leet = palabra_leet.replace(original, reemplazo)
    candidatos.add(palabra_leet)
    candidatos.add(palabra_leet.capitalize())

# 3. Adición de sufijos numéricos y símbolos frecuentes
sufijos = ['123', '2025', '2026', '!', '1234', '_*']
nuevos = set()
for cand in candidatos:
    for suf in sufijos:
        nuevos.add(cand + suf)
    candidatos.update(nuevos)

return list(candidatos)

```

```

def ataque_diccionario_heuristico(hash_objetivo, wordlist_base):
    """
    Simula el proceso de comparación iterativa Hash(w_i) == H(p)
    utilizando un diccionario base y reglas heurísticas.
    """

    print("[*] Iniciando simulación de ataque de diccionario con
    heurísticas...")

    intentos_totales = 0

    for palabra_base in wordlist_base:
        # Generar mutaciones inteligentes basadas en la palabra base
        candidatos_mutados = aplicar_heurísticas(palabra_base)

        for candidato in candidatos_mutados:
            intentos_totales += 1

            # Se calcula la función hash (en este ejemplo SHA-256 para
            demostración)
            hash_calculado = hashlib.sha256(candidato.encode('utf-
            8')).hexdigest()

            # Verificación de coincidencia matemática H(w_i) == H(p)
            if hash_calculado == hash_objetivo:
                print(f"[+] ¡Coincidencia encontrada exitosamente!")
                print(f"    - Palabra base del diccionario:
                '{palabra_base}'")
                print(f"    - Contraseña exacta descubierta: '{candidato}'")
                print(f"    - Intentos computacionales realizados:
                {intentos_totales}")
                return candidato

    print("[-] El ataque finalizó: la contraseña no se encontró en el
    diccionario evaluado.")

    return None

```

```
# --- Bloque de Ejecución Experimental ---

if __name__ == "__main__": # Definimos una contraseña objetivo simulada de
baja/mediana entropía humana contraseña_objetivo = "password123"

# Generamos su hash objetivo (representando el hash almacenado en un sistema
vulnerable) hash_objetivo = hashlib.sha256(contrasena_objetivo.encode('utf-
8')).hexdigest()

# Diccionario base simulado (palabras comunes de uso cotidiano) wordlist_base
= ["admin", "sistema", "password", "network", "user"]

# Ejecutar el modelo de ataque ataque_diccionario_heuristico(hash_objetivo,
wordlist_base)
```

Los experimentos ejecutados en el entorno controlado de laboratorio permitieron evidenciar la brecha crítica existente entre la solidez teórica de los algoritmos criptográficos modernos y las vulnerabilidades derivadas de su implementación operativa y del factor humano (Anderson, 2020; Stallings, 2017).

Rendimiento y Sensibilidad del Espacio de Claves en Ataques de Fuerza Bruta

Durante la ejecución del criptoanálisis exhaustivo mediante John the Ripper sobre los contenedores protegidos con cifrado PKZIP, se midió de forma empírica el impacto directo de la longitud del espacio de claves sobre el tiempo de resolución (Solar Designer, 2023).

Escenario A (Clave corta de 5 caracteres numéricos - 12345): El motor de procesamiento paralelizado completó la validación del espacio de búsqueda en un intervalo inferior a un segundo, alcanzando tasas superiores a 3.5×10^5 intentos por segundo (p/s). Esto confirma que las credenciales de baja entropía son computacionalmente inviables en entornos de producción actuales (Bishop, 2018).

Escenario B (Clave extendida de 9 caracteres - 123451234): Al incrementar el espacio de permutación, el tiempo de procesamiento aumentó de manera exponencial, evidenciando la propiedad defensiva de la longitud de clave recomendada por estándares internacionales (National Institute of Standards and Technology [NIST], 2020; Stallings, 2017).

Análisis Estructural del Cifrado Simétrico y Derivación de Claves (OpenSSL / KPA)

La experimentación en AlmaLinux utilizando OpenSSL demostró que la seguridad de un mensaje cifrado mediante AES-256-CBC no recae exclusivamente en la fortaleza del algoritmo simétrico, sino en el mecanismo de derivación de claves (Ferguson et al., 2010; Katz & Lindell, 2020):

Al ejecutar el cifrado sin parámetros explícitos de derivación robusta, la biblioteca emitió la advertencia correspondiente (WARNING: deprecated key derivation used), exponiendo a los sistemas heredados a ataques de precomputación (Paar & Pelzl, 2009).

La inspección hexadecimal a través de la utilidad xxd permitió visualizar el bloque inicial Salted__ (5361 6c74 6564 5f5f), comprobando la inserción de una cadena aleatoria (Salt) por usuario. Este mecanismo invalida de forma directa el uso de tablas arcoíris (Rainbow Tables) y ataques de diccionario estáticos precalculados (Biryukov et al., 2016; O'Flynn & Chen, 2014).

Impacto de la Predictibilidad Humana en los Ataques de Diccionario

Los resultados del modelado heurístico y la simulación de ataques de diccionario corroboraron que el comportamiento humano tiende a repetir patrones predecibles (nombres, fechas, secuencias de teclado y sustituciones leet) (Bonneau et al., 2012). Al contrastar el ataque de fuerza bruta pura frente al ataque de diccionario asistido por listas históricas (RockYou), se demostró que el uso de heurísticas reduce el espacio de búsqueda en varios órdenes de magnitud, permitiendo comprometer credenciales empresariales en segundos si las organizaciones omiten políticas estrictas de complejidad y rotación (Steube, 2022; Whitman & Mattord, 2021).

CONCLUSIONES

Vulnerabilidad Perimetral en PyMES: La investigación demostró que las brechas de seguridad en las pequeñas y medianas empresas rara vez provienen de fallas matemáticas en los algoritmos criptográficos modernos (como AES), sino de errores humanos en la elección de credenciales y de configuraciones obsoletas en los protocolos de transporte y almacenamiento (Anderson, 2020; Stallings, 2017).

Transferencia Tecnológica Educativa: La implementación práctica de laboratorios basados en software de código abierto (OpenSSL, John the Ripper, CrypTool 2) constituye una estrategia didáctica de alto valor para democratizar el conocimiento criptográfico, permitiendo que comunidades académicas y



sectores productivos vulnerables identifiquen y mitiguen riesgos cibernéticos con recursos limitados (Gate Learn, 2023; O'Flynn & Chen, 2014).

Defensa en Profundidad: Para proteger el tejido empresarial frente a ataques de fuerza bruta, diccionario y texto conocido (KPA), se concluye que es indispensable adoptar una arquitectura de defensa en profundidad que integre funciones de derivación lentas y resistentes a memoria (Argon2, bcrypt), políticas de frases de paso de alta entropía (passphrases) y controles de limitación de tasa (rate limiting) en los accesos perimetrales (Biryukov et al., 2016; NIST, 2020).

REFERENCIAS BIBLIOGRAFICAS

Anderson, R. (2020). Security engineering: A guide to building dependable distributed systems (3rd ed.). Wiley.

Biryukov, A., Dinu, D., & Khovratovich, D. (2016). Argon2: New generation of memory-hard functions for password hashing and other applications. En *2016 IEEE Symposium on Security and Privacy (SP)* (pp. 292–302). IEEE.

<https://www.computer.org/csdl/proceedingsarticle/eurosp/2016/07467361/12OmNvEyR5N>

Bishop, M. (2018). Computer security: Art and science (2nd ed.). Addison-Wesley Professional.

Boneh, D., & Shoup, V. (2023). A graduate course in applied cryptography (Version 0.6). Stanford University.

Bonneau, J., Herley, C., van Oorschot, P. C., & Stajano, F. (2012). The quest to replace passwords: A framework for evaluation of web authentication schemes. En *2012 IEEE Symposium on Security and Privacy* (pp. 553–567). IEEE. <https://doi.org/10.1109/SP.2012.44>

Ferguson, N., Schneier, B., & Kohno, T. (2010). Cryptography engineering: Design principles and practical applications. Wiley.

Gate Learn. (2023, 23 de mayo). Known-plaintext attacks (KPA) explained. Gate.io.<https://www.startupdefense.io/cyberattacks/known-plaintext-attack#1-introduction-to-known-plaintext-attack>

Katz, J., & Lindell, Y. (2020). Introduction to modern cryptography (3rd ed.). CRC Press. <https://www.routledge.com/Introduction-to-Modern-Cryptography-Revised-Third-Edition/Katz-Lindell/p/book/9781032496795>



- National Institute of Standards and Technology. (2020). Digital identity guidelines: Authentication and lifecycle management (NIST Special Publication 800-63B). U.S. Department of Commerce <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf>
- O'Flynn, C., & Chen, Z. D. (2014). ChipWhisperer: An open-source platform for hardware security research. En *Proceedings of the 2014 ACM Workshop on Information Security Applications* (pp. 243–260). Association for Computing Machinery. https://link.springer.com/chapter/10.1007/978-3-319-10175-0_17
- Paar, C., & Pelzl, J. (2009). Understanding cryptography: A textbook for students and practitioners. Springer. <https://cointhinktank.com/upload/crypto2/Understanding%20Cryptography%20by%20Christof%20Paar%20.pdf>
- Solar Designer. (2023). John the Ripper password cracker: Features and user manual. Openwall. <https://www.openwall.com/john/>
- Stallings, W. (2017). Cryptography and network security: Principles and practice (7th ed.). Pearson. <https://www.pearson.com/en-us/subject-catalog/p/cryptography-and-network-security-principles-and-practice/P200000003477?view=educator>
- Steube, J. (2022). Hashcat: Advanced password recovery. Hashcat Project. <https://hashcat.net/hashcat/>
- Whitman, M. E., & Mattord, H. J. (2021). Principles of information security (7th ed.). Cengage Learning. <https://faculty.cengage.com/works/9780357506431>