



Universidad Central del Ecuador
Facultad de Ingeniería y Ciencias Aplicadas
Ingeniería en Computación



Nombres:

- Antony Arguello
- David Fuelata
- Juan Jumbo
- Josue Masabanda
- Doménica Vizcarra

Curso: Octavo Semestre

Fecha: 05 - 06 – 2023

Materia: Criptografía y Seguridad de la Información

Resultados

Tabla de resultados

Algoritmo	#Palabras	Segundos				
		T-E1	T-E2	T-E3	T-E4	T-Total
Chacha - 20	10	0,000524	0,00164	0,0158	0,00202	0,004996
	100	0,00267	0,00161	0,00464	0,00572	0,00366
	1000	0,0014	0,00451	0,01	0,0018	0,004275
	10000	0,016	0,000293	0,0266	0,00877	0,01291575
	100000	0,0098	0,000298	0,0242	0,00244	0,0091845
	1000000	0,0098	0,00157	0,0266	0,00877	0,011685
AES	10	0,001978635	0,000274181	0,0022	0,00225687	0,001677422
	100	0,000422477	0,002122402	0,001965523	0,000978946	0,001372337
	1000	0,002530336	0,00176811	0,002353191	0,002087355	0,002184748
	10000	0,002684832	0,001782656	0,002686977	0,002284765	0,002359808
	100000	0,01793742	0,000260114	0,00572299	0,005020856	0,007235345
	1000000	0,083240337	0,002209186	0,009591341	0,005092382	0,025033312
Diffie - Hellman	10	0,00056	0,00909	No encripta, ni desencrypta		0,004825
	100	0,00216	0,00569			0,003925
	1000	0,00268	0,0093			0,00599
	10000	0,00364	0,01166			0,00765
	100000	0,0106	0,008766			0,00968
	1000000	0,0287	0,0107			0,0197
SHA - 512	10	0,00249	0,000112	No encripta, ni desencrypta		0,0026
	100	0,00293	0,000107			0,00151
	1000	0,00194	0,00013			0,00103
	10000	0,00375	0,000381			0,00206
	100000	0,00895	0,00316			0,00605
	1000000	0,248	0,0259			0,0253

T-E1	Tiempo de lectura de archivos
T-E2	Tiempo de generación de claves
T-E3	Tiempo de cifrado
T-E4	Tiempo de descifrado

Algoritmo Simétrico: Chacha 20

Con 10 Palabras

Tiempo N°1

```
▶ start_time1 = time()
book = Path("lorem10.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encryption_time1))

b'Lorem ipsum dolor sit amet consectetur adipisicing elit. Fugit, aliquid?'
Tiempo de lectura del archivo: 0.0005247592926025391 segundos
```

Tiempo N°2

```
[5] #Cifrado chacha-20 claves
start_time2 = time()
key = get_random_bytes(32) # Clave de 256 bits (32 bytes)
nonce = get_random_bytes(16)
print(key,nonce)
encryption_time2 = time() - start_time2
print("Tiempo de generación de claves: {} segundos".format(encryption_time2))

b'q\x0c\xbd$fx4\xc7:\xc2\x1a\xde\x1d\xa7\xbb\x17\xecy2\xaf\xee5\xad\x0e\xc6(+`x1e\xfd?[\xaa'
Tiempo de generación de claves: 0.0016405582427978516 segundos
```

Tiempo N°3

```
▶ #Cifrar texto
start_time3 = time()
cipher = Cipher(algorithms.ChaCha20(key, nonce), mode=None)
encryptor = cipher.encryptor()
ciphertext = encryptor.update(book) + encryptor.finalize()
print(ciphertext.hex())
encryption_time3 = time() - start_time3
print("Tiempo de cifrado: {} segundos".format(encryption_time3))

📄 5432a28152bbfc25ffc415b7fad9741761c6ddbc11088b101dbd37a26b69f7101e22d82e58ad520a1167b94c
Tiempo de cifrado: 0.0158236026763916 segundos
```

Tiempo N°4

```
▶ #Decifrar texto
start_time4 = time()
cipher = Cipher(algorithms.ChaCha20(key, nonce), mode=None)
decryptor = cipher.decryptor()
plaintext = decryptor.update(ciphertext) + decryptor.finalize()
print(plaintext.decode())
encryption_time4 = time() - start_time4
print("Tiempo de descifrado: {} segundos".format(encryption_time4))
```

↳ Lorem ipsum dolor sit amet consectetur adipisicing elit. Fugit, aliquid?
Tiempo de descifrado: 0.002027750015258789 segundos

Con 100 Palabras

Tiempo N°1

```
▶ start_time1 = time()
book = Path("lorem12.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encryption_time1))
```

b'Lorem, ipsum dolor sit amet consectetur adipisicing elit. Beatae, consectetur explicabo?
Tiempo de lectura del archivo: 0.0026733875274658203 segundos

Tiempo N°2

```
[5] #Cifrado chacha-20 claves
start_time2 = time()
key = get_random_bytes(32) # Clave de 256 bits (32 bytes)
nonce = get_random_bytes(16)
print(key, nonce)
encryption_time2 = time() - start_time2
print("Tiempo de generación de claves: {} segundos".format(encryption_time2))
```

b'b=\xc3Q\x7f\x03\xcb\x9a\xab\xfb5b\x80\xac\xa4\xd7N0\xedV\xe5\x1fyN1\xa5\xcbU>\xe70\x06]'
Tiempo de generación de claves: 0.0016109943389892578 segundos

Tiempo N°3

```
6 #Cifrar texto
start_time3 = time()
cipher = Cipher(algorithms.ChaCha20(key, nonce), mode=None)
encryptor = cipher.encryptor()
ciphertext = encryptor.update(book) + encryptor.finalize()
print(ciphertext.hex())
encryption_time3 = time() - start_time3
print("Tiempo de cifrado: {} segundos".format(encryption_time3))
```

↳ 33282fee24b83a3a4193a47998bf37e7cb5a2a465616bb6c211f9f4677fef4fcb271ddbbaa3ade0746428d2
Tiempo de cifrado: 0.004648447036743164 segundos

Tiempo N°4

```
[7] #Decifrar texto
start_time4 = time()
cipher = Cipher(algorithms.ChaCha20(key, nonce), mode=None)
decryptor = cipher.decryptor()
plaintext = decryptor.update(ciphertext) + decryptor.finalize()
print(plaintext.decode())
encryption_time4 = time() - start_time4
print("Tiempo de descifrado: {} segundos".format(encryption_time4))

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Beatae, consectetur explicabo?
Tiempo de descifrado: 0.00572967529296875 segundos
```

Con 1000 Palabras

Tiempo N°1

```
start_time1 = time()
book = Path("lorem13.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encryption_time1))

b'Lorem ipsum, dolor sit amet consectetur adipisicing elit. Nisi, voluptatibus.
Tiempo de lectura del archivo: 0.0014028549194335938 segundos
```

Tiempo N°2

```
[5] #Cifrado chacha-20 claves
start_time2 = time()
key = get_random_bytes(32) # Clave de 256 bits (32 bytes)
nonce = get_random_bytes(16)
print(key,nonce)
encryption_time2 = time() - start_time2
print("Tiempo de generación de claves: {} segundos".format(encryption_time2))

b'\xa5\xe0\r\xeb\xae\xb8\x7f\xd4\xd6\xabyn\xc5E\xcd\xc3\x0b%\xcf\xe08\r\x90\xf3)\xd7\xfa0\xf5\x9fT\xf3'
Tiempo de generación de claves: 0.0045108795166015625 segundos
```

Tiempo N°3

```
#Cifrar texto
start_time3 = time()
cipher = Cipher(algorithms.ChaCha20(key, nonce), mode=None)
encryptor = cipher.encryptor()
ciphertext = encryptor.update(book) + encryptor.finalize()
print(ciphertext.hex())
encryption_time3 = time() - start_time3
print("Tiempo de cifrado: {} segundos".format(encryption_time3))
```

5641c454c0881bfde49a3ca876338e1fa2e02c9d932adafdf91992618d02ac9e32a789b4b98102d9c865c
Tiempo de cifrado: 0.010084390640258789 segundos

Tiempo N°4

```
[7] #Decifrar texto
start_time4 = time()
cipher = Cipher(algorithms.ChaCha20(key, nonce), mode=None)
decryptor = cipher.decryptor()
plaintext = decryptor.update(ciphertext) + decryptor.finalize()
print(plaintext.decode())
encryption_time4 = time() - start_time4
print("Tiempo de descifrado: {} segundos".format(encryption_time4))
```

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Nisi, voluptatibus.
Tiempo de descifrado: 0.0018095970153808594 segundos

Con 10000 Palabras

Tiempo N°1

```
start_time1 = time()
book = Path("lorem14.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encryption_time1))
```

b'Lorem ipsum dolor sit amet consectetur adipisicing elit. Necessitatibus similique,
Tiempo de lectura del archivo: 0.01600813865661621 segundos

Tiempo N°2

```
#Cifrado chacha-20 claves
start_time2 = time()
key = get_random_bytes(32) # Clave de 256 bits (32 bytes)
nonce = get_random_bytes(16)
print(key, nonce)
encryption_time2 = time() - start_time2
print("Tiempo de generación de claves: {} segundos".format(encryption_time2))
```

b'\x12~>\x05\x85`\xf7\x8f?\xf8TK\x8f\xcc9%\xc1\xdfp4D\xbfW\x8d\xd7K\xdc\xe0\x9eE\xa5i'
Tiempo de generación de claves: 0.0002932548522949219 segundos

Tiempo N°3

```
#Cifrar texto
start_time3 = time()
cipher = Cipher(algorithms.ChaCha20(key, nonce), mode=None)
encryptor = cipher.encryptor()
ciphertext = encryptor.update(book) + encryptor.finalize()
print(ciphertext.hex())
encryption_time3 = time() - start_time3
print("Tiempo de cifrado: {} segundos".format(encryption_time3))
```

➤ e0815b0b9e05af86e1cf136c120bb330d727e1ff5463725fc5e898d7880d91296ee7bcf9
Tiempo de cifrado: 0.02666163444519043 segundos

Tiempo N°4

```
#Decifrar texto
start_time4 = time()
cipher = Cipher(algorithms.ChaCha20(key, nonce), mode=None)
decryptor = cipher.decryptor()
plaintext = decryptor.update(ciphertext) + decryptor.finalize()
print(plaintext.decode())
encryption_time4 = time() - start_time4
print("Tiempo de descifrado: {} segundos".format(encryption_time4))
```

➤ Lorem ipsum dolor sit amet consectetur adipisicing elit. Necessitatibus similique,
Tiempo de descifrado: 0.008775472640991211 segundos

Con 100000 Palabras

Tiempo N°1

```
start_time1 = time()
book = Path("lorem15.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encryption_time1))
```

b'Lorem ipsum dolor sit amet consectetur adipisicing elit. Optio minima perspiciatis
Tiempo de lectura del archivo: 0.00980687141418457 segundos

Tiempo N°2

```
[5] #Cifrado chacha-20 claves
start_time2 = time()
key = get_random_bytes(32) # Clave de 256 bits (32 bytes)
nonce = get_random_bytes(16)
print(key,nonce)
encryption_time2 = time() - start_time2
print("Tiempo de generación de claves: {} segundos".format(encryption_time2))

b''gK\x7f\xff\xd7\xf8\x9b\xff\x14\xcb\x85Z0\xce/\xbaT\xe5\x98\x0b1\\\x10:\x8d\x0fK\
Tiempo de generación de claves: 0.0002980232238769531 segundos
```

Tiempo N°3

```
[6] #Cifrar texto
start_time3 = time()
cipher = Cipher(algorithms.ChaCha20(key, nonce), mode=None)
encryptor = cipher.encryptor()
ciphertext = encryptor.update(book) + encryptor.finalize()
print(ciphertext.hex())
encryption_time3 = time() - start_time3
print("Tiempo de cifrado: {} segundos".format(encryption_time3))

533fe5cb54a4fcaeac52c63b1ce69ca38e97405cbf14d3f20c415cf65c16244d4e93ed21f4d28
Tiempo de cifrado: 0.024291038513183594 segundos
```


Tiempo N°4

```
[7] #Decifrar texto
start_time4 = time()
cipher = Cipher(algorithms.ChaCha20(key, nonce), mode=None)
decryptor = cipher.decryptor()
plaintext = decryptor.update(ciphertext) + decryptor.finalize()
print(plaintext.decode())
encryption_time4 = time() - start_time4
print("Tiempo de descifrado: {} segundos".format(encryption_time4))

Lorem ipsum dolor sit amet consectetur adipisicing elit. Optio minima perspiciatis
Tiempo de descifrado: 0.002440929412841797 segundos
```

Con 1000000 Palabras

Tiempo N°1

```
 start_time1 = time()
book = Path("lorem16.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encryption_time1))

b'Lorem ipsum dolor sit amet consectetur adipisicing elit. Optio minima perspiciatis
Tiempo de lectura del archivo: 0.00980687141418457 segundos
```

Tiempo N°2

```
[5] #Cifrado chacha-20 claves
start_time2 = time()
key = get_random_bytes(32) # Clave de 256 bits (32 bytes)
nonce = get_random_bytes(16)
print(key,nonce)
encryption_time2 = time() - start_time2
print("Tiempo de generación de claves: {} segundos".format(encryption_time2))

b'\xcd\xfa\x15>\x96\x86\x1c\x11\xf3oK;\xbd{\xc5\xc6*\xc4Q\x0bf\xe8q\x90\xd5\xe4$
Tiempo de generación de claves: 0.001577138900756836 segundos
```

Tiempo N°3

```
#Cifrar texto
start_time3 = time()
cipher = Cipher(algorithms.ChaCha20(key, nonce), mode=None)
encryptor = cipher.encryptor()
ciphertext = encryptor.update(book) + encryptor.finalize()
print(ciphertext.hex())
encryption_time3 = time() - start_time3
print("Tiempo de cifrado: {} segundos".format(encryption_time3))

e0815b0b9e05af86e1cf136c120bb330d727e1ff5463725fc5e898d7880d91296ee7bcf9
Tiempo de cifrado: 0.02666163444519043 segundos
```

Tiempo N°4

```
#Decifrar texto
start_time4 = time()
cipher = Cipher(algorithms.ChaCha20(key, nonce), mode=None)
decryptor = cipher.decryptor()
plaintext = decryptor.update(ciphertext) + decryptor.finalize()
print(plaintext.decode())
encryption_time4 = time() - start_time4
print("Tiempo de descifrado: {} segundos".format(encryption_time4))

Lorem ipsum dolor sit amet consectetur adipisicing elit. Necessitatibus similique,
Tiempo de descifrado: 0.008775472640991211 segundos
```


Algoritmo Simétrico: AES

Con 10 Palabras

Tiempo N°1

```
start_time1 = time()
book = Path("lorem10.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encryption_time1))
```

```
b'Lorem ipsum dolor sit amet consectetur adipisicing elit. Fugit, aliquid?'
Tiempo de lectura del archivo: 0.001978635787963867 segundos
```

Tiempo N°2

```
start_time1 = time()
key = get_random_bytes(16) # Clave de 16 bytes (128 bits)
iv = get_random_bytes(AES.block_size)
print(key,iv)
encryption_time1 = time() - start_time1
print("Tiempo de generación de claves: {} segundos".format(encryption_time1))
```

```
b'\x9e\xef\xcJOL\xf9K\xbf\xc5\xd\xd2K\x03' b'\x03.\xc8^VQ\xda\xfa\r\x82\xa0\xf1\xe3\xc2\x99\xdf'
Tiempo de generación de claves: 0.0002741813659667969 segundos
```

Tiempo N°3

```
start_time1 = time()
cipher = AES.new(key, AES.MODE_CBC, iv)
ciphertext = cipher.encrypt(pad(book, AES.block_size))
print(book.hex())
encryption_time1 = time() - start_time1
print("Tiempo de cifrado del archivo: {} segundos".format(encryption_time1))
```

```
4c6f72656d20697073756d20646f6c6f722073697420616d657420636f6e736563746574757220616469706973
Tiempo de cifrado del archivo: 0.0022394657135009766 segundos
```

Tiempo N°4

```
start_time1 = time()
cipher = AES.new(key, AES.MODE_CBC, iv)
book = unpad(cipher.decrypt(ciphertext), AES.block_size)
print(book.decode())
encryption_time1 = time() - start_time1
print("Tiempo de descifrado del archivo: {} segundos".format(encryption_time1))
```

```
↳ Lorem ipsum dolor sit amet consectetur adipisicing elit. Fugit, aliquid?
Tiempo de descifrado del archivo: 0.0022568702697753906 segundos
```

Con 100 Palabras

Tiempo N°1

```
start_time1 = time()
book = Path("lorem12.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encryption_time1))
```

b'Lorem, ipsum dolor sit amet consectetur adipisicing elit. Beatae, consectetur explicabo? I

Tiempo de lectura del archivo: 0.0004224772216796875 segundos

Tiempo N°2

```
start_time1 = time()
key = get_random_bytes(16) # Clave de 16 bytes (128 bits)
iv = get_random_bytes(AES.block_size)
print(key,iv)
encryption_time1 = time() - start_time1
print("Tiempo de generación de claves: {} segundos".format(encryption_time1))
```

b'8-\xaa\x19s\xd0{L\xc3|\xa4a\x93gp@' b'\x06\x96\x9b\xa8p\x18~\xafr\x8f\n&\xd5\x07\xd6\xfa'

Tiempo de generación de claves: 0.0021224021911621094 segundos

Tiempo N°3

```
start_time1 = time()
cipher = AES.new(key, AES.MODE_CBC, iv)
ciphertext = cipher.encrypt(pad(book, AES.block_size))
print(book.hex())
encryption_time1 = time() - start_time1
print("Tiempo de cifrado del archivo: {} segundos".format(encryption_time1))
```

4c6f72656d2c20697073756d20646f6c6f722073697420616d657420636f6e73656374657475722061646970

Tiempo de cifrado del archivo: 0.0019655227661132812 segundos

Tiempo N°4

```
start_time1 = time()
cipher = AES.new(key, AES.MODE_CBC, iv)
book = unpad(cipher.decrypt(ciphertext), AES.block_size)
print(book.decode())
encryption_time1 = time() - start_time1
print("Tiempo de descifrado del archivo: {} segundos".format(encryption_time1))
```

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Beatae, consectetur explicabo? I

Tiempo de descifrado del archivo: 0.0009789466857910156 segundos

Con 1000 Palabras

Tiempo N°1

```
start_time1 = time()
book = Path("lorem13.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encryption_time1))
```

b'Lorem ipsum, dolor sit amet consectetur adipisicing elit. Nisi, voluptatibus. Illo p
Tiempo de lectura del archivo: 0.002530336380004883 segundos

Tiempo N°2

```
start_time1 = time()
key = get_random_bytes(16) # Clave de 16 bytes (128 bits)
iv = get_random_bytes(AES.block_size)
print(key,iv)
encryption_time1 = time() - start_time1
print("Tiempo de generación de claves: {} segundos".format(encryption_time1))
```

b'\x15\xab\xac\xae\x85\x82 \wy\xc9\x0f|\xbf' b'H\x8a\xaa[\xcd\x17]aL\xb2\xf9:\xe1<\xe5a'
Tiempo de generación de claves: 0.0017681121826171875 segundos

Tiempo N°3

```
start_time1 = time()
cipher = AES.new(key, AES.MODE_CBC, iv)
ciphertext = cipher.encrypt(pad(book, AES.block_size))
print(book.hex())
encryption_time1 = time() - start_time1
print("Tiempo de cifrado del archivo: {} segundos".format(encryption_time1))
```

4c6f72656d20697073756d2c20646f6c6f722073697420616d657420636f6e7365637465747572206
Tiempo de cifrado del archivo: 0.002353191375732422 segundos

Tiempo N°4

```
start_time1 = time()
cipher = AES.new(key, AES.MODE_CBC, iv)
book = unpad(cipher.decrypt(ciphertext), AES.block_size)
print(book.decode())
encryption_time1 = time() - start_time1
print("Tiempo de descifrado del archivo: {} segundos".format(encryption_time1))
```

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Nisi, voluptatibus. Illo pers
Tiempo de descifrado del archivo: 0.0020873546600341797 segundos

Con 10000 Palabras

Tiempo N°1

```
start_time1 = time()
book = Path("lorem14.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encryption_time1))
```

b'Lorem ipsum dolor sit amet consectetur adipisicing elit. Necessitatibus similique, enim p
Tiempo de lectura del archivo: 0.0026848316192626953 segundos

Tiempo N°2

```
start_time1 = time()
key = get_random_bytes(16) # Clave de 16 bytes (128 bits)
iv = get_random_bytes(AES.block_size)
print(key,iv)
encryption_time1 = time() - start_time1
print("Tiempo de generaci3n de claves: {} segundos".format(encryption_time1))
```

b'\x1b\xec\xd1\x9f\x88\xcf\xaf\xaf\x05\xb6\x1bk\xfbjt\x9c' b'\x044\x02\xf5\xd9\x1aB\xbdM`C\xcfVy']'
Tiempo de generaci3n de claves: 0.0017826557159423828 segundos

Tiempo N°3

```
start_time1 = time()
cipher = AES.new(key, AES.MODE_CBC, iv)
ciphertext = cipher.encrypt(pad(book, AES.block_size))
print(book.hex())
encryption_time1 = time() - start_time1
print("Tiempo de cifrado del archivo: {} segundos".format(encryption_time1))
```

4c6f72656d20697073756d20646f6c6f722073697420616d657420636f6e7365637465747572206164697069736
Tiempo de cifrado del archivo: 0.0026869773864746094 segundos

Tiempo N°4

```
start_time1 = time()
cipher = AES.new(key, AES.MODE_CBC, iv)
book = unpad(cipher.decrypt(ciphertext), AES.block_size)
print(book.decode())
encryption_time1 = time() - start_time1
print("Tiempo de descifrado del archivo: {} segundos".format(encryption_time1))
```

Lorem ipsum dolor sit amet consectetur adipisicing elit. Necessitatibus similique, enim prov
Tiempo de descifrado del archivo: 0.0022847652435302734 segundos

Con 100000 Palabras

Tiempo N°1

```
start_time1 = time()
book = Path("lorem15.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encryption_time1))
```

b'Lorem ipsum dolor sit amet consectetur adipisicing elit. Optio minima perspiciatis fugit aperiam mo
Tiempo de lectura del archivo: 0.017937421798706055 segundos

Tiempo N°2

```
start_time1 = time()
key = get_random_bytes(16) # Clave de 16 bytes (128 bits)
iv = get_random_bytes(AES.block_size)
print(key,iv)
encryption_time1 = time() - start_time1
print("Tiempo de generación de claves: {} segundos".format(encryption_time1))

b"\xb8b\x83'\xa0\xe3\x8c\xde\x9c\x88\xb0M\xb1\xc0b\xec\xe7" b'^\xca\xcb\xd0\xea\x10_\x9c\x'
Tiempo de generación de claves: 0.0002601146697998047 segundos
```

Tiempo N°3

```
start_time1 = time()
cipher = AES.new(key, AES.MODE_CBC, iv)
ciphertext = cipher.encrypt(pad(book, AES.block_size))
print(book.hex())
encryption_time1 = time() - start_time1
print("Tiempo de cifrado del archivo: {} segundos".format(encryption_time1))
```

Tiempo de cifrado del archivo: 0.005722999572753906 segundos

Tiempo N°4

```
[22] start_time1 = time()
cipher = AES.new(key, AES.MODE_CBC, iv)
book = unpad(cipher.decrypt(ciphertext), AES.block_size)
print(book.decode())
encryption_time1 = time() - start_time1
print("Tiempo de descifrado del archivo: {} segundos".format(encryption_time1))
```

Lorem ipsum dolor sit amet consectetur adipisicing elit. Optio minima perspiciatis fugit aperiam mollitia porro quam quis des
Tiempo de descifrado del archivo: 0.005020856857299805 segundos

Con 1000000 Palabras

Tiempo N°1

```
start_time1 = time()
book = Path("lorem16.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encryption_time1))
```

o>Lorem ipsum dolor sit amet consectetur adipisicing elit. Optio minima perspiciat
Tiempo de lectura del archivo: 0.013240337371826172 segundos

Tiempo N°2

```
start_time1 = time()
key = get_random_bytes(16) # Clave de 16 bytes (128 bits)
iv = get_random_bytes(AES.block_size)
print(key,iv)
encryption_time1 = time() - start_time1
print("Tiempo de generación de claves: {} segundos".format(encryption_time1))
```

b'\xf9J\x95\xb0\x05;JA\xe6\x96\xd5\x11I\x8c\x08c' b"\xa6'\x1c39\xec\x11>\x00\xe6\xcdA^\xde\x0e\xd8"
Tiempo de generación de claves: 0.002209186553955078 segundos

Tiempo N°3

```
start_time1 = time()
cipher = AES.new(key, AES.MODE_CBC, iv)
ciphertext = cipher.encrypt(pad(book, AES.block_size))
print(book.hex())
encryption_time1 = time() - start_time1
print("Tiempo de cifrado del archivo: {} segundos".format(encryption_time1))
```

Tiempo de cifrado del archivo: 0.009591341018676758 segundos

Tiempo N°4

```
start_time1 = time()
cipher = AES.new(key, AES.MODE_CBC, iv)
book = unpad(cipher.decrypt(ciphertext), AES.block_size)
print(book.decode())
encryption_time1 = time() - start_time1
print("Tiempo de descifrado del archivo: {} segundos".format(encryption_time1))
```

Lorem ipsum dolor sit amet consectetur adipisicing elit. Optio minima perspiciatis fu
Tiempo de descifrado del archivo: 0.0050923824310302734 segundos

Algoritmo Asimétrico: Diffie Hellman

Con 10 Palabras

Tiempo N°1

```
start_time1 = time()
book = Path("lorem10.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encryption_time1))
```

b'Lorem ipsum dolor sit amet consectetur adipisicing elit. Fugit, aliquid?'
Tiempo de lectura del archivo: 0.0005373954772949219 segundos

Con 100 Palabras

Tiempo N°1

```
start_time1 = time()
book = Path("lorem12.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encryption_time1))
```

b'Lorem, ipsum dolor sit amet consectetur adipisicing elit. Beat
Tiempo de lectura del archivo: 0.0024123191833496094 segundos

Con 1000 Palabras

Tiempo N°1

```
start_time1 = time()
book = Path("lorem13.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encryption_time1))
```

b'Lorem ipsum, dolor sit amet consectetur adipisicing elit. Nis
Tiempo de lectura del archivo: 0.002755403518676758 segundos

Con 10000 Palabras

Tiempo N°1

```

start_time1 = time()
book = Path("lorem14.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encry

b'Lorem ipsum dolor sit amet consectetur adipisicing elit. Nec
Tiempo de lectura del archivo: 0.003473043441772461 segundos

```

Con 100000 Palabras

Tiempo N°1

```

start_time1 = time()
book = Path("lorem15.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encry

b'Lorem ipsum dolor sit amet consectetur adipisicing elit. Opti
Tiempo de lectura del archivo: 0.014160633087158203 segundos

```

Tiempo N°2

```

print("\nTiempo de lectura del archivo: {} segundos".format(encry

Clave publica Alice 76981320339
Clave privada Alice 74346447037
Clave publica Bob 36718834521
Clave privada Bob 32109519968
Tiempo de lectura del archivo: 0.007436037063598633 segundos

```

Tiempo N°3 y Tiempo N°4

Justificación

El algoritmo de Diffie-Hellman no se utiliza directamente para encriptar mensajes, sino para establecer una clave compartida entre dos partes que luego puede utilizarse para encriptar y desencriptar mensajes con un algoritmo de cifrado simétrico, como AES (Advanced Encryption Standard).

Función Hash: SHA_512

Con 10 Palabras

Tiempo N°1

```
start_time1 = time()
book = Path("lorem10.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encryp
```

b'Lorem ipsum dolor sit amet consectetur adipisicing elit. Fugit
Tiempo de lectura del archivo: 0.00038170814514160156 segundos

Tiempo N°2

Mensaje: b'Lorem ipsum dolor sit amet consectetur adipisicing e
Hash SHA-512: eb864f3607eb9ce0510b32b0f25dbc15f01b34917f5d89f81
Tiempo de lectura del archivo: 0.0003046989440917969 segundos

Con 100 Palabras

Tiempo N°1

```
start_time1 = time()
book = Path("lorem12.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encry
```

b'Lorem, ipsum dolor sit amet consectetur adipisicing elit. Bea
Tiempo de lectura del archivo: 0.0020096302032470703 segundos

Tiempo N°2

Mensaje: b'Lorem, ipsum dolor sit amet consectetur adipisicing
Hash SHA-512: bb7a34623f9ff327721a0aae0e29849d17110904745732de
Tiempo de lectura del archivo: 0.0002231597900390625 segundos

Con 1000 Palabras

Tiempo N°1

```
start_time1 = time()
book = Path("lorem13.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encryption_time1))

b'Lorem ipsum, dolor sit amet consectetur adipisicing elit. Nisi, quod,
Tiempo de lectura del archivo: 0.002185344696044922 segundos
```

Tiempo N°2

```
Mensaje: b'Lorem ipsum, dolor sit amet consectetur adipisicing elit. Nisi, quod,
Hash SHA-512: 57e4ac8eacccb5bc1cc163f437446ac0b0acbf62aa3a8781e5
Tiempo de lectura del archivo: 0.0003898143768310547 segundos
```

Con 10000 Palabras

Tiempo N°1

```
start_time1 = time()
book = Path("lorem14.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encryption_time1))

b'Lorem ipsum dolor sit amet consectetur adipisicing elit. Necessitatibus,
Tiempo de lectura del archivo: 0.003266572952270508 segundos
```

Tiempo N°2

```
Mensaje: b'Lorem ipsum dolor sit amet consectetur adipisicing elit. Necessitatibus,
Hash SHA-512: 7e4f848febdff0beff478e3f4c2379844c0ce9a84295a0d752
Tiempo de lectura del archivo: 0.0013113021850585938 segundos
```

Con 100000 Palabras

Tiempo N°1

```
start_time1 = time()
book = Path("lorem15.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encryp

b'Lorem ipsum dolor sit amet consectetur adipisicing elit. Optio
Tiempo de lectura del archivo: 0.009742975234985352 segundos
```

Tiempo N°2

```
Mensaje: b'Lorem ipsum dolor sit amet consectetur adipisicing eli
Hash SHA-512: 17a1fd7bd162ecf23ef0c2dec2ac766761cb0b2ea0ecfa2770
Tiempo de lectura del archivo: 0.009775638580322266 segundos
```

Con 1000000 Palabras

Tiempo N°1

```
start_time1 = time()
book = Path("lorem15.txt").read_text().encode("utf-8")
print(book)
encryption_time1 = time() - start_time1
print("Tiempo de lectura del archivo: {} segundos".format(encryp

b'Lorem ipsum dolor sit amet consectetur adipisicing elit. Optio
Tiempo de lectura del archivo: 0.016053438186645508 segundos
```

Tiempo N°2

```
Mensaje: b'Lorem ipsum dolor sit amet consectetur adipisicing eli
Hash SHA-512: 17a1fd7bd162ecf23ef0c2dec2ac766761cb0b2ea0ecfa2
Tiempo de lectura del archivo: 0.0183870792388916 segundos
```

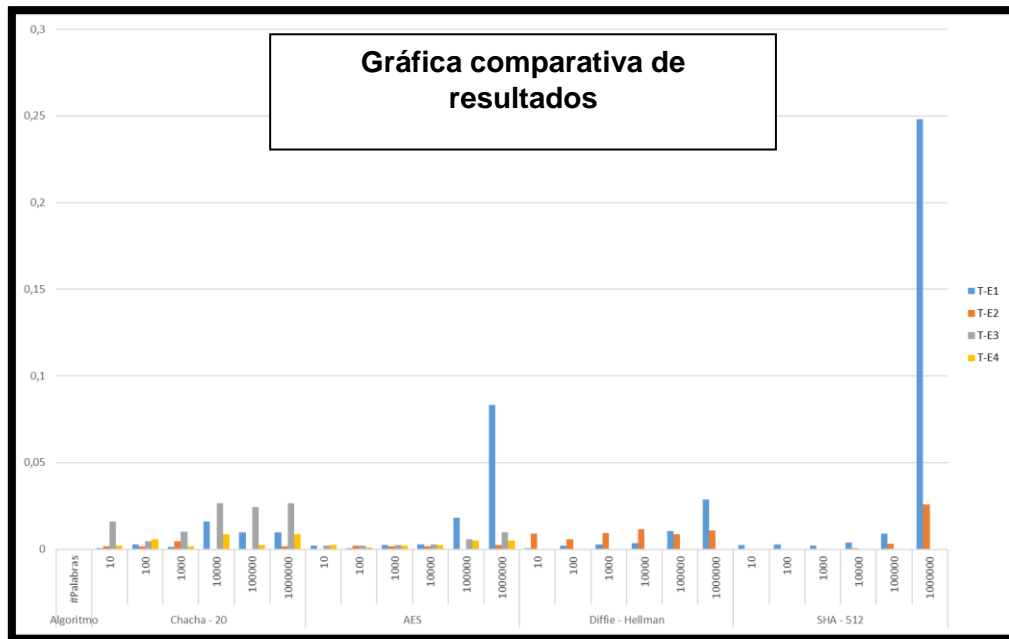
Tiempo N°3 y Tiempo N°4

Justificación

Las funciones hash criptográficas están diseñadas para ser unidireccionales, lo que significa que no se pueden revertir fácilmente. No están destinadas al descryptado, sino a generar un resumen único y fijo de los datos de entrada.

Cuando se utiliza una función hash criptográfica, como SHA-512, para almacenar contraseñas o verificar la integridad de los datos, generalmente no se puede descryptar el hash para obtener la información original. En cambio, se compara el hash calculado de los datos de entrada con el hash almacenado previamente. Si los hashes coinciden, se considera que los datos son auténticos o las contraseñas son iguales.

Conclusiones



1. Comparando el tiempo de generación de claves, cifrado y descifrado de los algoritmos simétricos AES y ChaCha20:

- AES (Advanced Encryption Standard): Es ampliamente utilizado y considerado seguro. Sin embargo, el proceso de generación de claves puede ser relativamente lento, especialmente para claves de mayor longitud. El cifrado y descifrado con AES son rápidos y eficientes, lo que lo hace adecuado para aplicaciones que requieren un alto rendimiento.
- ChaCha20: Es un algoritmo de cifrado de flujo diseñado para ser rápido y seguro. El tiempo de generación de claves en ChaCha20 es más rápido que en AES, lo que lo hace atractivo para aplicaciones que requieren una generación rápida de claves. El cifrado y descifrado con ChaCha20 también son rápidos y eficientes.

En conclusión, en términos de generación de claves, ChaCha20 ofrece un tiempo más rápido que AES. Sin embargo, la elección entre AES y ChaCha20 dependerá de otros factores, como el nivel de seguridad requerido y el entorno de aplicación.

2. Hablando sobre los algoritmos Diffie-Hellman y SHA-512 y su tiempo para generar claves:

- Diffie-Hellman: Es un algoritmo de intercambio de claves utilizado en criptografía de clave pública. La generación de claves en Diffie-Hellman puede ser computacionalmente costosa, especialmente cuando se utilizan parámetros de clave largos. El tiempo de generación de claves aumenta con la longitud de los parámetros, lo que puede hacer que sea más lento en comparación con otros algoritmos criptográficos.
- SHA-512: Es un algoritmo de hash seguro que produce una salida de 512 bits. Sin embargo, no se utiliza para generar claves directamente. SHA-512 se

emplea más comúnmente en aplicaciones como la integridad de datos y la autenticación, donde no está involucrada la generación de claves.

En conclusión, Diffie-Hellman puede requerir más tiempo para generar claves en comparación con otros algoritmos criptográficos. SHA-512, por otro lado, no se utiliza directamente para la generación de claves, sino para otras aplicaciones criptográficas.

3. Coste computacional al realizar la generación de claves con los algoritmos AES, ChaCha20, Diffie-Hellman y SHA-512:
 - AES: El coste computacional de generación de claves en AES puede variar según la longitud de la clave deseada. En general, AES puede ser más exigente en términos de recursos computacionales en comparación con otros algoritmos criptográficos debido a la complejidad de su diseño y operaciones.
 - ChaCha20: Es conocido por su eficiencia computacional y su bajo coste en términos de recursos. La generación de claves en ChaCha20 puede ser menos costosa en comparación con AES.
 - Diffie-Hellman: El coste computacional de generación de claves en Diffie-Hellman puede ser alto, especialmente cuando se utilizan parámetros de clave largos. El proceso de cálculo de las claves compartidas en Diffie-Hellman implica operaciones matemáticas intensivas.
 - SHA-512: En el caso de SHA-512, no se utiliza directamente para la generación de claves, por lo que no está directamente relacionado con el coste computacional de generar claves.

En resumen, en términos de coste computacional, AES puede requerir más recursos que ChaCha20, y Diffie-Hellman puede ser más costoso en comparación con AES y ChaCha20 debido a sus operaciones matemáticas intensivas. Sin embargo, es importante tener en cuenta que el coste computacional puede variar según la implementación específica de los algoritmos y las características del hardware utilizado.

Bibliografía

- Boneh, D., & Shoup, V. (2020). A Graduate Course in Applied Cryptography. <https://crypto.stanford.edu/~dabo/cryptobook/>
- Rogaway, P. (2004). Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. Cryptology ePrint Archive, Report 2004/035. <https://eprint.iacr.org/2004/035.pdf>
- National Institute of Standards and Technology (NIST). (2015). Secure Hash Standard (SHS) (FIPS PUB 180-4). <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>