



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Pablito y sus amigos

Seguridad Térmica:
Algoritmos RSA y ECC
para la Comunicación
entre Sensores de
Temperatura

Criptografía
TEL252 - 2 - CSSJ

Índice

1. Resumen	3
2. Introducción	3
3. Método	4
3.1. Problema o Hipótesis	4
3.1.1. Análisis	4
3.1.2. Resultados	4
4. Experimentación	4
4.1. Prueba de Concepto	4
4.2. Análisis de resultados	5
5. Discusión	5
6. Conclusiones	6
7. Anexos	6
8. Referencias	6

1. Resumen

Este proyecto se enfoca en la creciente preocupación sobre cómo la avanzada capacidad de procesamiento de la computación cuántica puede comprometer la efectividad de los algoritmos de criptografía asimétrica actuales, como RSA y ECC. A través de una metodología cuantitativa que incluye revisión de literatura, análisis matemáticos y simulaciones computacionales, evaluamos la vulnerabilidad de estos algoritmos frente a los problemas presentados por el centro meteorológico ACME, el cual a través del análisis de la temperatura recabada por sensores en diversas ubicaciones de Chile ha presentado problemas de consistencia en sus predicciones generales y en la seguridad de la comunicación entre estos sensores.

2. Introducción

En la era digital, la criptografía desempeña un papel crucial en la protección de la información. Su evolución ha sido paralela a los avances tecnológicos, proporcionando seguridad en comunicaciones y transacciones en línea.

La computación cuántica emerge como un cambio de paradigma, prometiendo un procesamiento de datos exponencialmente más rápido. Esta evolución plantea nuevos retos y oportunidades en el campo de la seguridad informática.

La capacidad mejorada de las computadoras cuánticas podría hacer vulnerables los algoritmos de criptografía asimétrica actuales, como RSA y ECC, esenciales para la seguridad de Internet.

Este proyecto busca evaluar cómo la combinación y complementación entre estos 2 algoritmos puede garantizar la seguridad en el cifrado y descifrado de mensajes entre sensores de temperatura como los del centro meteorológico ACME.

3. Método

3.1. Problema o Hipótesis

- La creciente capacidad de procesamiento cuántico amenaza la seguridad de los algoritmos de criptografía asimétrica, que son la base de la seguridad en Internet.
- A través de la investigación y desarrollo de cifrado cuántico, se pueden crear soluciones seguras y eficientes para proteger la información en la era cuántica.

3.1.1. Análisis

RSA

- Base Teórica: Se fundamenta en la dificultad de factorizar el producto de dos números primos grandes.
- Implementación: Utilizado en una amplia gama de aplicaciones de seguridad, desde comunicaciones seguras hasta firmas digitales.

ECC

- Base Teórica: Se basa en el problema del logaritmo discreto en curvas elípticas.
- Implementación: Eficiente en términos de tamaño de clave y rendimiento, adecuado para dispositivos con recursos limitados.

3.1.2. Resultados

RSA

- Eficacia: Ha demostrado ser altamente efectivo en el cifrado de mensajes, manteniendo la confidencialidad y la integridad de la información.
- Consideraciones de Rendimiento: Más lento y requiere más recursos que ECC para un nivel de seguridad equivalente.

ECC

- Eficacia: Proporciona un cifrado robusto y seguro con un menor tamaño de clave.
- Beneficios de Rendimiento: Más rápido y eficiente que RSA, ideal para aplicaciones móviles y de IoT.

4. Experimentación

A continuación, se presenta el funcionamiento de nuestro proyecto:

```

Terminal - julian@mx: ~/Documentos/proyecto cripto vfinal
Archivo Editar Ver Terminal Pestañas Ayuda
julian@mx:~/Documentos/proyecto cripto vfinal
$ python3 proyecto.py
RSA Cifrado: 0fa42f2ea7fa68da4da884cca8fd67a8ebb198d6a8ee060879a9091c7b260737a22626d505eeab3782548e1
0767a9b845e12a1c26adc6405e56c51d68f47a0bf418bee6f47f7ba3a30d42bccf499e66ee2ed530d84b6027799db603a561
50df8735a80fcd86a5a98f6810fee79f7ee3aff491ed506600ca550fa9a81d823223b66ff6ffda411a499c899b293e913ce
c6b4dc2e7df0dcd29ac3e0f1b28e7724c5f517d3b04e5a8f77735acc2b3c8c8801adc57cb9c719cb73efb6fd2048fa5ce45
39ac5e58b202c6e7fd2149463348a0435c4429d97cc5958d91c2dd56f64d5d950f619eb02e72b38cfd05a28eb14ca648f93
2cf9886c1884d8909b8e4ad61
ECC Cifrado: ea6760e4e9ab391cf554977aecfba60b39fbbbc27247bbbe0d33d322d4ef0a7fce532aab7da52191690bb76
5db653c2501755d1346dbd1e36881fe62fa0f63723a89e2666441e17d404aedbc53c970a25d50640e532ba6181e615c9f0c
b6fc6317ac71e20cc5b2a25767eff81fa423ab7fa39be712553352361da39ea4e1a489eb4808f2baf5fcd6ccc34de0f2b2a
ac1fe6f42f0169d0f49385e8b3146f863b752534903f995420c78857b83013a7f4b09edd6c6d99dd492c395df90a16023a5a
3c3dbd8419ee489fd53540fce86ba4ab09503ade284d1168387fd4b236079750242da10c70e85369065b22257c2476f8c451
41efd2d0f6ac1d45ef6ef8acc
Texto Final Descifrado: b'La temperatura es de 25 grados celcius'
julian@mx:~/Documentos/proyecto-cripto-vfinal$

```

Figura 1: Ejecución del código

4.1. Prueba de Concepto

Se utilizó un doble cifrado con los algoritmos RSA y ECC, con el fin de obtener una mayor seguridad y confidencialidad en el cifrado del mensaje. Primero se cifra el mensaje con RSA y luego con ECC; para después descifrar mediante ECC y RSA respectivamente.

A continuación, se presenta el código en Python:

```

projecto.py x ecc.py
projecto.py > ...
1 from cryptography.hazmat.backends import default_backend
2 from cryptography.hazmat.primitives import serialization
3 from cryptography.hazmat.primitives.asymmetric import rsa, ec
4 from cryptography.hazmat.primitives import hashes, serialization
5 from cryptography.hazmat.primitives.asymmetric import padding
6 from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
7 from os import urandom
8
9 iv = urandom(16) # El tamaño del IV depende del algoritmo de cifrado utilizado
10
11 def generar_par_claves_rsa():
12     clave_privada = rsa.generate_private_key(
13         public_exponent=65537,
14         key_size=2048,
15         backend=default_backend()
16     )
17     clave_publica = clave_privada.public_key()
18     return clave_privada, clave_publica
19
20 def generar_par_claves_ecc():
21     clave_privada = ec.generate_private_key(
22         ec.SECP256R1(), # Puedes elegir una curva diferente si es necesario
23         backend=default_backend()
24     )
25     clave_publica = clave_privada.public_key()
26     return clave_privada, clave_publica

```

Figura 2: Funciones Generar claves RSA y ECC

En la figura 2, se observa que se utilizó la librería cryptography para las principales funciones del código; Se generan las claves públicas con `.public_key()`; y se generan las claves privadas con `.generate_private_key()`, se puede observar que el proceso de creación de clave privada es distinto según el algoritmo.

Ahora procederemos a analizar los cifrados de cada algoritmo.

```

3 def cifrar_rsa(mensaje, clave_publica):
4     texto_cifrado = clave_publica.encrypt(
5         mensaje,
6         padding.OAEP(
7             mgf=padding.MGF1(algorithm=hashes.SHA256()),
8             algorithm=hashes.SHA256(),
9             label=None
10        )
11    )
12    return texto_cifrado
13
14 def descifrar_rsa(texto_cifrado, clave_privada):
15     texto_plano = clave_privada.decrypt(
16         texto_cifrado,
17         padding.OAEP(
18             mgf=padding.MGF1(algorithm=hashes.SHA256()),
19             algorithm=hashes.SHA256(),
20             label=None
21        )
22    )
23    return texto_plano

```

Figura 3: Cifrado y descifrado RSA

En la figura 3 se cifra y descifra el mensaje con el algoritmo RSA, utilizando las funciones `.encrypt()` y `.decrypt()`, y dentro de estas se utiliza el algoritmo `hashes.SHA256()`.

```

def cifrar_ecc(mensaje, clave_publica, clave_privada):
    clave_compartida = clave_privada.exchange(ec.ECDH(), clave_publica)
    # Usa la clave compartida para derivar una clave simétrica y cifrar el mensaje.
    clave_compartida_bytes = int.from_bytes(clave_compartida, 'big').to_bytes(32, 'big') # Suponiendo un tamaño de clave de 256
    cifrador = Cipher(algorithms.AES(clave_compartida_bytes), modes.CFB(iv), backend=default_backend())
    cifrador_aes = cifrador.encryptor()
    texto_cifrado = cifrador_aes.update(mensaje) + cifrador_aes.finalize()
    return texto_cifrado

def descifrar_ecc(texto_cifrado, clave_privada, clave_publica):
    clave_compartida = clave_privada.exchange(ec.ECDH(), clave_publica)
    # Usa la clave compartida para derivar una clave simétrica y descifrar el mensaje.
    clave_compartida_bytes = int.from_bytes(clave_compartida, 'big').to_bytes(32, 'big') # Suponiendo un tamaño de clave de 256
    cifrador = Cipher(algorithms.AES(clave_compartida_bytes), modes.CFB(iv), backend=default_backend())
    cifrador_aes = cifrador.decryptor()
    texto_plano = cifrador_aes.update(texto_cifrado) + cifrador_aes.finalize()
    return texto_plano

```

Figura 4: Cifrado y descifrado ECC

En la figura 4 implementamos AES dentro del cifrado y descifrado de ECC, con el objetivo de poder compatibilizar el algoritmo RSA con ECC.

```
67
68 # Uso de ejemplo:
69 clave_privada_rsa, clave_publica_rsa = generar_par_claves_rsa()
70 clave_privada_ecc, clave_publica_ecc = generar_par_claves_ecc()
71
72 mensaje = b"La temperatura es de 25 grados celcius"
73
74 # Cifrar con RSA
75 texto_cifrado_rsa = cifrar_rsa(mensaje, clave_publica_rsa)
76 print(f"RSA Cifrado: {texto_cifrado_rsa.hex()}")
77
78 # Cifrar con ECC
79 texto_cifrado_ecc = cifrar_ecc(texto_cifrado_rsa, clave_publica_ecc, clave_privada_ecc)
80 print(f"ECC Cifrado: {texto_cifrado_ecc.hex()}")
81
82 # Descifrar con ECC
83 texto_cifrado_descifrado = descifrar_ecc(texto_cifrado_ecc, clave_privada_ecc, clave_publica_ecc)
84
85 # Descifrar con RSA
86 texto_final_descifrado = descifrar_rsa(texto_cifrado_descifrado, clave_privada_rsa)
87 print(f"Texto Final Descifrado: {texto_final_descifrado}")
88
89
90
```

Figura 5: Implementación de las funciones

En la figura 5 se implementan las funciones y los print por pantalla.

4.2 Análisis de resultados

En el análisis de los resultados de nuestro código, siempre obtuvimos una respuesta “correcta”, así que aquí pueden ver que si modificamos el mensaje, el cifrado y descifrado sigue funcionando bien:

```

Archivo Editar Ver Terminal Pestañas Ayuda
50df8735a80fcd86a5a98f6819fee79f7ee3aff491ed566609ca550fa9a81d823223b66ff6ffda411a499c899b293e913ce
c6b4dc2e7df0dcd29ac3e0f1b28e7724c5f517d3b04e5a8f77735acc2b3c8c8801adc57cb9c719cb73efb6fd2048fa5ce45
39ac5e58b202c6e7fd2149463348a0435c4429d97cc5958d91c2dd56f64d5d950f619eb02e72b38cf0d5a28eb14ca648f93
2cf9886c1884d8909b8e4ad61
ECC Cifrado: ea6760e4e9ab391cf554977aefba60b39fbbcc27247bbbe0d33d322d4ef0a7fce532aab7da52191690bb76
5db053c2501755d1346dbd1e36881fe62fa0f63723a89e2666441e17d404aedbc53c970a25d50640e532ba6181e615c9f0c
b0fc6317ac71e20cc5b2a2570eff81fa423ab7fa39bc712553352361da39ea4e1a489eb480bf2baf5fcd6ccc34de0f2b2a
ac1fe6f42f01000f4938eab314df903b75253a903f995420c78857b83013a7f14b09ddedcd99dd492c395df90a16023a5a
3c3dbd8419ee489fd53540fce86ba4ab09503ade284d118387fd4b236079750242da10c70e8536906522257c2476f8c451
41efd2d0f6acd1d5ef6ef8acc
Texto Final Descifrado: b'La temperatura es de 25 grados celcius'
julian@mx:~/Documentos/proyecto cripto vfinal
$ python3 proyecto.py
RSA Cifrado: 8f1636d6d8987a1e5253637ec8357ebfa041c631596d90ec4794ca748c2a6177066490e44718062453387
6b9609e9e9a0db1a0b7e915f5bbe2ab523dfe9f9dabbad0b060f5c581489fbd979007ad1db093f80f9952ce9fc33379
9bf4dd0dd4ee37a747ba97f50e9711e8b07119cb5360475e095381955d4515b238092365a577181b34b1bbad6eb6b5389b64
7d151d167a92f75bd04a47ed2e84d080d3f80190fb2c8bf63f95863c41feea872be7d1d2558574f9225920dd6396187b28c
8d78end36a3faa26f3fed0f3aba8c1d7d8506a8d46b23bf7469d82eeba63390cce4c75758ed744a02f776e9f3ca30a679404
b243771ec278ba4a90c359cf5
ECC Cifrado: ae464964b0452458b37f5d344343dcac725769ce0d4d1bd27c853741c3a83b9e42ddce84934e1fcd3a36c94
b8794ab580be263a2d8a3cc9b1d672010eab3aeb764a02ee92f7ee6002407b22fedb81f9274ae9f9effb060e03e4e3562d48
9c4746d2cc98399e220907cc8c0197350f6487203e1780d29f92a9ca70fc8889b5342340c7fa65a6c578b6dfbbae23550
76b13fed7d7235b71c74ce199bd212c0bf17a0a275de74c63009d9f7a6d0f53497ad640e3c1a1dce7082a1e54667013a7e
a1d0cc1e356646e7f5a4841644aa439ac9843d3340776f711dc6cb12700a49526f37017d87572ee7e8361e790e566689df74
a30f8d8287b045b57417423f2
Texto Final Descifrado: b'La temperatura es de 10 grados celcius'
julian@mx:~/Documentos/proyecto cripto vfinal
$

```

5. Discusión

Como grupo creemos que queda pendiente la mejora del código enfocada en las llaves, ya que solo nos enfocamos en el cifrado y descifrado de un mensaje, pero no en la seguridad y protección de las llaves. Por otro lado, creemos que el código cumple a cabalidad con su función y cumple con solucionar el problema inicial, ya que el cifrado del mensaje entre los sensores fue un éxito.

6. Conclusiones

La combinación de algoritmos de cifrado como RSA y ECC puede ofrecer una estrategia viable para mejorar la seguridad de los datos. La fusión de estos sistemas criptográficos puede fortalecer la resistencia frente a posibles ataques y mitigar las debilidades individuales de cada uno.

No obstante, la implementación exitosa de esta combinación requiere un diseño adaptado a las circunstancias y una gestión cuidadosa de las claves, ya que puede aumentar la complejidad del sistema. A pesar de las ventajas, la seguridad de cualquier sistema criptográfico no solo depende de los algoritmos utilizados, sino también de su implementación adecuada y de las prácticas seguras de gestión de claves.

Para concluir, podemos decir que la combinación de RSA y ECC puede ser una estrategia efectiva para reforzar la seguridad de la información, pero en su implementación se debe prestar atención a la gestión de claves y a la correcta aplicación de los algoritmos para lograr un sistema sólido y resistente a posibles amenazas.

7. Anexos

- Como retroalimentación proporcionada por nuestra profesora del curso Berioska Contreras, sería buena idea aumentar la seguridad de las llaves incluyendo algún otro algoritmo como lo puede ser AES, ya que en nuestra propuesta de proyecto nos enfocamos principalmente en mejorar y solucionar el cifrado del mensaje a través de los sensores. Además, pudimos experimentar de mejor manera y realizar más pruebas con los algoritmos utilizados, cómo pudo ser el caso dado en que se cifrase el mensaje primero con ECC para posteriormente utilizar RSA, caso el cual no se nos ocurrió probar dado ciertos inconvenientes a la hora de manejar los códigos.
- Como retroalimentación recibida por el grupo “Codekeepers”, la explicación fue clara y ofreció una visión equilibrada sobre las ventajas y consideraciones al fusionar RSA y ECC para mejorar la seguridad de los datos. Sin embargo, una consideración importante podría ser explorar la concatenación de llaves de cifrado, que puede introducir complejidades adicionales en la gestión, especialmente en términos de la generación, distribución y almacenamiento seguro de los datos. Además, sería útil investigar los posibles riesgos de seguridad asociados. Esto incluiría identificar posibles vulnerabilidades que podrían surgir debido a la interacción de múltiples algoritmos y cómo mitigar estos.

- Como retroalimentación dada para el grupo “Codekeepers”, encontramos que su propuesta fue muy original y única, contando con una excelente presentación por parte del equipo y abarcando un tema muy interesante como lo es la transición de A5/1 a A5/3. Fue el único grupo dónde se explicó incluso en pizarra como funcionaban los distintos algoritmos que se abarcan en Kazumi, lo cual es a destacar, y en general no encontramos muchos reparos en la presentación más que ciertos aspectos técnicos como la explicación que se hizo del código en video, la cual a nuestro parecer si bien fue correcta pudo tener un mejor apartado de sonido, o quizás haber implementado más claramente el tema de los sensores de temperaturas presentado en clases, pero son detalles que en general no afectan mucho a lo que se quiso exponer y es un tema amplio que se mostró de muy buena manera.

8. Referencias

Paar C. and Pelzl J. (2010). Understanding Cryptography. Springer.