



**LABORATORIUM PEMBELAJARAN ILMU KOMPUTER**  
**FAKULTAS ILMU KOMPUTER**  
**UNIVERSITAS BRAWIJAYA**

---

BAB	: KRIPTOGRAFI
NAMA	: MUHAMMAD FARISY CHANIAGO
NIM	: 215150200111038
TANGGAL	: 31/05/2023
ASISTEN	: GABRIELLE EVAN FARREL

---

## PROSEDUR PERCOBAAN

### A. Enkripsi/Dekripsi

1. Install library pyCryptodome

```
pip install PyCryptodome
# pengguna Endeavor/ArchLinux dapat pula menggunakan pacman
sudo pacman -S python-pycryptodome
```

2. Jalankan kode berikut:

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad

key = b'KunciRahasiaSaya'
cipher = AES.new(key, AES.MODE_ECB)

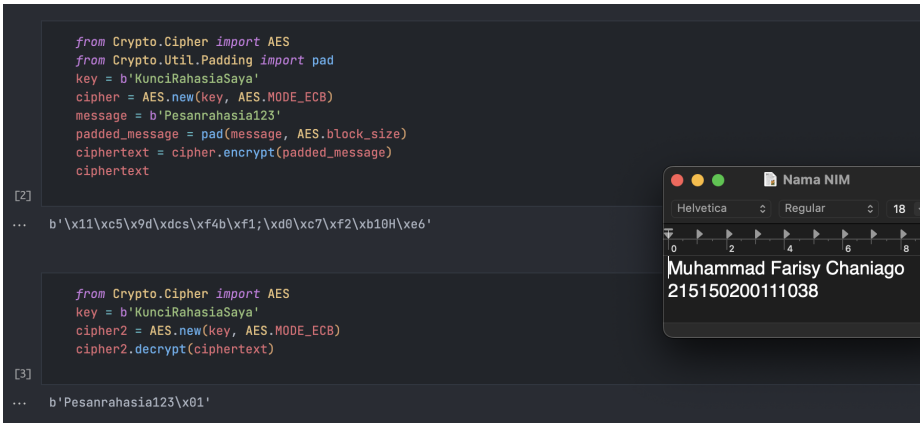
message = b'Pesanrahasia123'
padded_message = pad(message, AES.block_size)
ciphertext = cipher.encrypt(padded_message)
ciphertext
```

3. Jalankan kode berikut

```
from Crypto.Cipher import AES

key = b'KunciRahasiaSaya'
cipher2 = AES.new(key, AES.MODE_ECB)
cipher2.decrypt(ciphertext)
```

## Penjelasan output



The screenshot shows a Jupyter Notebook with two code cells. The first cell, labeled [2], contains Python code for AES encryption. It imports AES from Crypto.Cipher and padding from Crypto.Util.Padding. It defines a key 'KunciRahasiaSaya', creates an AES cipher object with the key and AES.MODE\_ECB, pads the message 'Pesanrahasia123' to the block size, and encrypts it. The output of this cell is a hexadecimal string: `b'\x11\xc5\x9d\xdc\xfb\xfb;\xd0\xc7\xf2\xb10H\xe6'`. The second cell, labeled [3], contains Python code for AES decryption. It imports AES from Crypto.Cipher and defines the same key. It creates an AES cipher object, decrypts the ciphertext, and removes the padding. The output of this cell is the original message: `b'Pesanrahasia123\x01'`. A small window titled 'Nama NIM' is visible in the background, showing the name 'Muhammad Farisy Chaniago' and the NIM number '215150200111038'.

Kode diatas melakukan enkripsi menggunakan algoritma Advanced Encryption Standard (AES). Algoritma Advanced Encryption Standard (AES) adalah salah satu algoritma enkripsi simetris yang digunakan secara luas untuk mengamankan data. AES merupakan standar enkripsi yang telah diadopsi oleh pemerintah Amerika Serikat dan digunakan secara global.

Setelah eksekusi kode awal, variabel ciphertext akan berisi teks terenkripsi yang dihasilkan dari pesan yang telah dienkripsi dengan kunci rahasia menggunakan algoritma AES dengan mode ECB. Output yang dihasilkan merupakan isi dari pesan yang sudah terenkripsi.

Kita juga dapat mengembalikan isi pesan dengan melakukan dekripsi dari data yang telah terenkripsi dengan menggunakan kunci(key) yang sama.

4. Pada langkah 3, gantilah nilai key = 'KunciRahasia1234'. Apakah error yang Anda dapatkan?

## Penjelasan output

```
from Crypto.Cipher import AES
key = b'KunciRahasia1234'
cipher2 = AES.new(key, AES.MODE_ECB)
cipher2.decrypt(ciphertext)

b"cl\xa1vI'%\x9dY\xfc\xcd\x9ahS\xca\xe7"
```

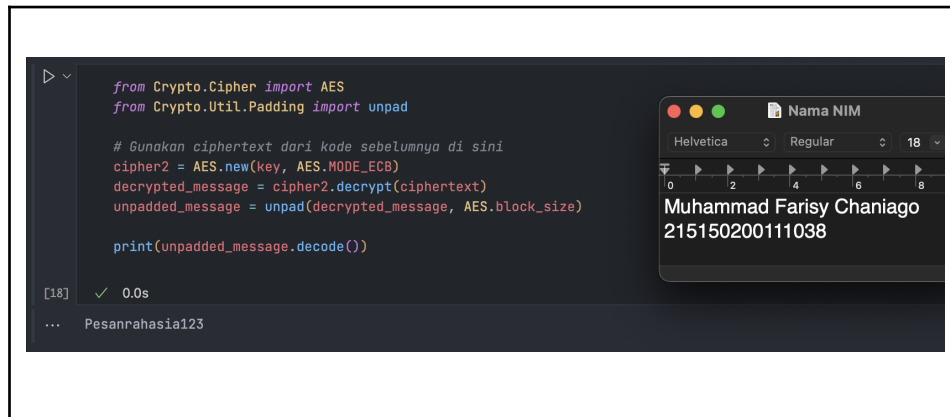
Ketika kita mengganti nilai key yang digunakan untuk melakukan dekripsi, hasil dekripsi tidak sesuai dengan pesan awal saat sebelum dilakukan enkripsi. Hal ini terjadi karena key yang kita gunakan untuk dekripsi tidak sama dengan key yang kita gunakan saat enkripsi. Tetapi karena panjang dari key masih sama, tidak terjadi error saat dijalankan, hanya isi pesan yang didapatkan berbeda.

5. Koreksilah hasil error pada Nomor 4 dengan menggunakan rekomendasi hasil search engine yang biasa Anda gunakan. . Screen capture kode hasil koreksi dan hasilnya.

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad
key = b'KunciRahasiaSaya'
cipher2 = AES.new(key, AES.MODE_ECB)
unpadded_message = unpad(cipher2.decrypt(ciphertext), AES.block_size)
print(unpadded_message)

[21] ✓ 0.0s
... b'Pesannahasia123'
```

6. Koreksilah hasil error pada Nomor 4 dengan menggunakan hasil rekomendasi ChatGPT. . Screen capture kode hasil koreksi dan hasilnya.



```
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad

# Gunakan ciphertext dari kode sebelumnya di sini
cipher2 = AES.new(key, AES.MODE_ECB)
decrypted_message = cipher2.decrypt(ciphertext)
unpadded_message = unpad(decrypted_message, AES.block_size)

print(unpadded_message.decode())
```

[18] ✓ 0.0s

... Pesanrahasia123

7. Amati hasil dari output langkah 5 dan 6, apakah yang dapat disimpulkan?

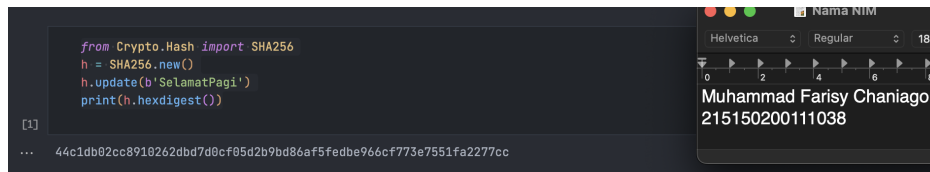
Kesimpulan yang didapatkan adalah koreksi hasil error yang direkomendasikan oleh ChatGPT lebih lengkap jika dibandingkan dengan hasil dari search engine google. Pada ChatGPT kode juga menggunakan method decode() untuk mendapatkan inti dari pesan.

## B. Hashing

1. Jalankan kode berikut

```
from Crypto.Hash import SHA256
h = SHA256.new()
h.update(b'SelamatPagi')
print(h.hexdigest())
```

Penjelasan output

A screenshot of a Python terminal window. The code being executed is: 

```
from Crypto.Hash import SHA256
h = SHA256.new()
h.update(b'SelamatPagi')
print(h.hexdigest())
```

The output shown is a long hexadecimal string: `44c1db02cc8910262dbd7d0cf05d2b9bd86af5fedbe966cf773e7551fa2277cc`. The terminal window has a title bar that says "Nama NIM" and a font menu showing "Helvetica", "Regular", and "18".

Kode diatas menggunakan library "Crypto" untuk mengimpor kelas "SHA256" dari modul "Hash". Library "Crypto" adalah modul yang digunakan untuk operasi kriptografi dalam Python.

SHA (Secure Hash Algorithm) adalah keluarga algoritma hash kriptografi yang digunakan untuk menghasilkan hash atau nilai hash dari data yang diberikan. Algoritma-algoritma dalam keluarga SHA dikembangkan oleh National Security Agency (NSA) dan diterbitkan oleh National Institute of Standards and Technology (NIST) di Amerika Serikat.

Hash merupakan representasi numerik atau nilai tetap yang unik yang dihasilkan dari data input yang berukuran apa pun. Hash digunakan untuk mengidentifikasi atau memverifikasi integritas data, di mana perubahan kecil pada data input akan menghasilkan hash yang berbeda.

Setelah mengimpor kelas "SHA256", kode tersebut membuat objek baru yang disebut "h" dengan memanggil konstruktor "SHA256.new()". Objek ini akan digunakan untuk menghitung hash SHA-256 dari data yang diberikan.

Kemudian, metode "update()" dipanggil pada objek "h" dengan parameter berupa data yang akan di-hash. Dalam kasus ini, data yang di-hash adalah byte-string "SelamatPagi" (huruf 'S' dan 'P' harus menggunakan huruf besar untuk memastikan bahwa itu adalah byte-string).

Akhirnya, metode "hexdigest()" dipanggil pada objek "h" untuk menghasilkan representasi dalam bentuk string dari hash yang

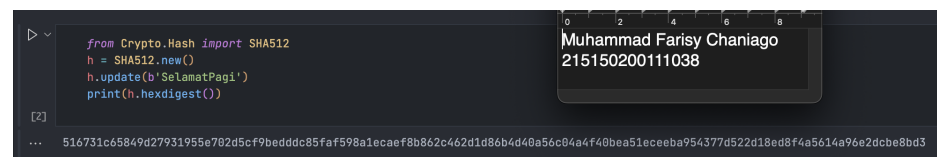
dihasilkan. Nilai yang dikembalikan dari metode ini dicetak menggunakan fungsi "print()".

Jadi, kode tersebut akan menghasilkan dan mencetak hash SHA-256 dari byte-string "SelamatPagi".

## 2. Jalankan kode berikut

```
from Crypto.Hash import SHA512
h = SHA512.new()
h.update(b'SelamatPagi')
print(h.hexdigest())
```

### Penjelasan output



Kode ini merupakan kode yang sama dengan kode pada langkah sebelumnya, perbedaannya hanya pada metode hash yang digunakan, yaitu SHA512. Karena menggunakan SHA512, hasil hash yang didapatkan lebih panjang.

Perbedaannya :

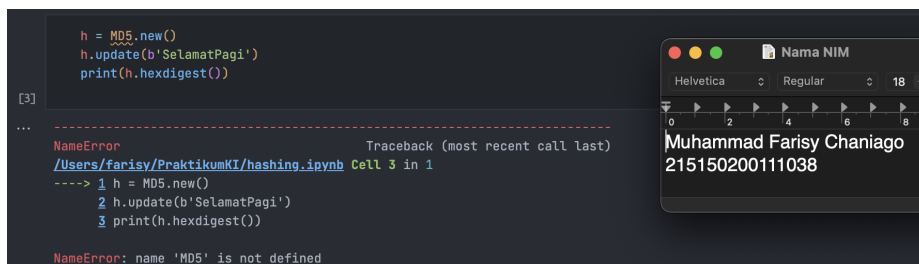
1. Ukuran Output Hash:
  - SHA-512 menghasilkan hash dengan panjang 512 bit atau 64 byte.
  - SHA-256 menghasilkan hash dengan panjang 256 bit atau 32 byte.
2. Panjang Blok:
  - SHA-512 menggunakan panjang blok 1024 bit.

- SHA-256 menggunakan panjang blok 512 bit.
3. Keamanan:
- Karena ukuran output yang lebih besar, SHA-512 dianggap lebih kuat dan tahan terhadap serangan brute-force dibandingkan dengan SHA-256.
  - Namun, dalam banyak kasus, keamanan SHA-256 sudah cukup untuk keperluan praktis.
4. Kinerja:
- Karena panjang blok yang lebih besar, SHA-512 dapat membutuhkan lebih banyak waktu dan sumber daya komputasi untuk menghasilkan hash dibandingkan dengan SHA-256.

3. Jalankan kode berikut

```
h = MD5.new()  
h.update(b'SelamatPagi')  
print(h.hexdigest())
```

Penjelasan output

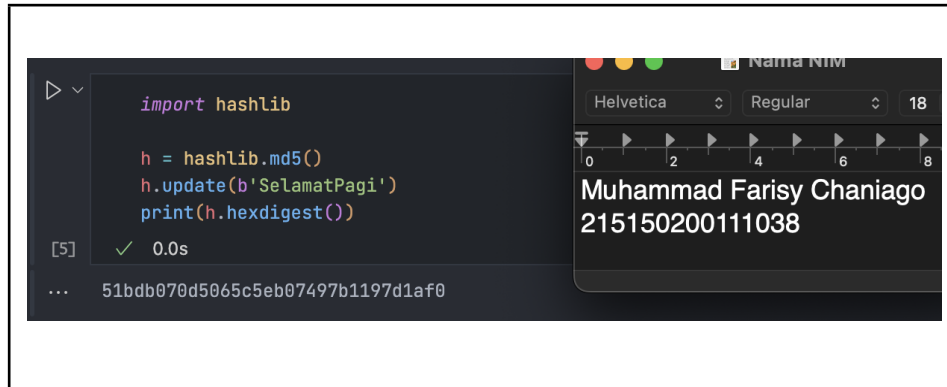


```
h = MD5.new()  
h.update(b'SelamatPagi')  
print(h.hexdigest())  
[3]  
...  
NameError                                Traceback (most recent call last)  
/Users/faris/PraktikumKI/hashing.ipynb Cell 3 in 1  
----> 1 h = MD5.new()  
      2 h.update(b'SelamatPagi')  
      3 print(h.hexdigest())  
  
NameError: name 'MD5' is not defined
```

Nama NIM  
Helvetica Regular 18  
Muhammad Farisy Chaniago  
215150200111038

Ketika kode dijalankan, maka akan terjadi error karena “MD5” belum didefinisikan.

4. Koreksilah hasil error pada Nomor 3 dengan menggunakan rekomendasi hasil search engine yang biasa Anda gunakan. Screen capture kode hasil koreksi dan hasilnya.



The screenshot shows a Python terminal window with the following code and output:

```
import hashlib

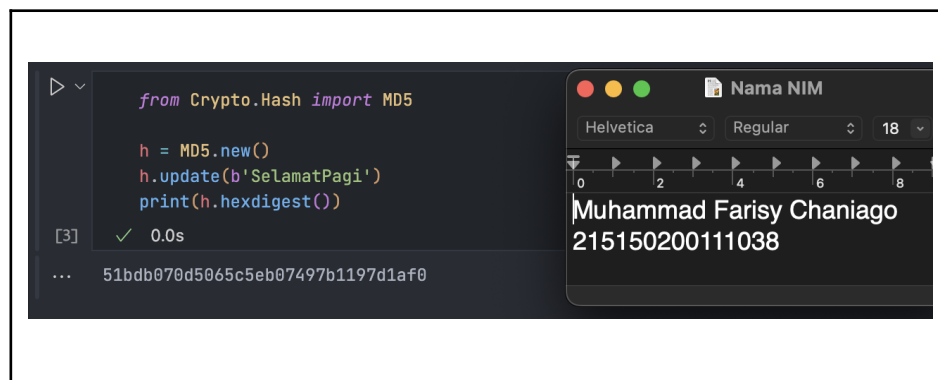
h = hashlib.md5()
h.update(b'SelamatPagi')
print(h.hexdigest())
```

[5] ✓ 0.0s

... 51bdb070d5065c5eb07497b1197d1af0

On the right, a text box displays the name and NIM: Muhammad Farisy Chaniago 215150200111038.

5. Koreksilah hasil error pada Nomor 3 dengan menggunakan hasil rekomendasi ChatGPT. . Screen capture kode hasil koreksi dan hasilnya.



The screenshot shows a Python terminal window with the following code and output:

```
from Crypto.Hash import MD5

h = MD5.new()
h.update(b'SelamatPagi')
print(h.hexdigest())
```

[3] ✓ 0.0s

... 51bdb070d5065c5eb07497b1197d1af0

On the right, a text box displays the name and NIM: Muhammad Farisy Chaniago 215150200111038.

6. Amati hasil dari output langkah 4 dan 5, apakah yang dapat disimpulkan?

Terdapat perbedaan dari solusi yang diberikan ChatGPT dan google search engine. Pada koreksi yang direkomendasi oleh google, menggunakan modul “hashlib”. Sedangkan, koreksi yang



direkomendasi oleh ChatGPT, menambahkan kode "from Crypto.Hash import MD5" agar "MD5" dapat digunakan. Walaupun memiliki penggunaan kode yang berbeda, hasil yang didapatkan tetap sama.

## C. Digital Signature

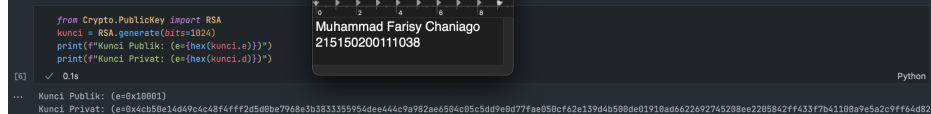
1. Jalankan kode berikut

```
from Crypto.PublicKey import RSA

kunci = RSA.generate(bits=1024)

print(f"Kunci Publik: (e={hex(kunci.e)})")
print(f"Kunci Privat: (e={hex(kunci.d)})")
```

Penjelasan output



```
from Crypto.PublicKey import RSA
kunci = RSA.generate(bits=1024)
print(f"Kunci Publik: (e={hex(kunci.e)})")
print(f"Kunci Privat: (e={hex(kunci.d)})")
```

[0] ✓ 0.1s Python

... Kunci Publik: (e=0x10001)  
Kunci Privat: (e=0x4c2b58e14d49c4c48f4ffff2d5db7968e3b3833355954dee444c9a982ae6504c85c5dd9e8d77fae850cf2e13944b580de01918ad6622692745208ee2205842ff433f7b4110ba9e5a2c9ff64d82)

Kode diatas menggunakan library "Crypto" untuk mengimpor kelas "RSA" dari modul "PublicKey". library "Crypto" adalah modul yang digunakan untuk operasi kriptografi dalam Python.

RSA (Rivest-Shamir-Adleman) adalah salah satu algoritma kriptografi asimetris yang paling populer dan banyak digunakan. Algoritma RSA

ditemukan oleh Ron Rivest, Adi Shamir, dan Leonard Adleman pada tahun 1977.

RSA menggunakan sepasang kunci, yaitu kunci publik (public key) dan kunci privat (private key), yang saling terkait secara matematis. Kunci publik digunakan untuk mengenkripsi pesan, sedangkan kunci privat digunakan untuk mendekripsi pesan yang dienkripsi dengan kunci publik.

Setelah mengimpor kelas "RSA", kode tersebut membuat objek baru yang disebut "kunci" dengan menggunakan metode "generate()" pada kelas "RSA". Metode ini digunakan untuk menghasilkan sepasang kunci RSA yang terdiri dari kunci publik dan kunci privat. Dalam contoh ini, kunci dengan panjang 1024 bit dihasilkan dengan parameter "bits=1024".

Selanjutnya, kode mencetak informasi tentang kunci yang dihasilkan. Baris pertama menggunakan f-string untuk mencetak nilai kunci publik dalam representasi heksadesimal dengan format "(e=...)", di mana "e" adalah eksponen kunci publik. Fungsi "hex()" digunakan untuk mengubah nilai eksponen menjadi heksadesimal.

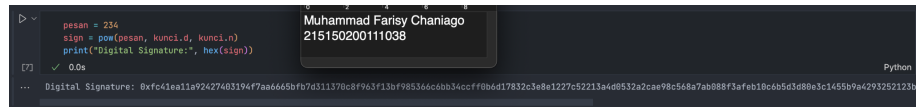
Baris kedua mencetak nilai kunci privat dalam representasi heksadesimal dengan format yang serupa seperti baris pertama.

Jadi, kode diatas akan menghasilkan dan mencetak kunci publik dan kunci privat dalam bentuk representasi heksadesimal.

## 2. Jalankan kode berikut

```
pesan = 234
sign = pow(pesan, kunci.d, kunci.n)
print("Digital Signature:", hex(sign))
```

## Penjelasan output



```
> pesan = 234
    sign = pow(pesan, kunci.d, kunci.n)
    print("Digital Signature:", hex(sign))

[7] ✓ 0.0s

... Digital Signature: 0xfc41ea11a92427403194f7aa6665bf7d311370c8f963f13bf985366c0bb34ccff0b6d17832c3e8e1227c52213a4d8532a2cae98c568a7ab088f3afeb18c4b5d3d88e3c1455b9a4293252123b

Muhammad Farisy Chaniago
215150200111038
```

Pertama, terdapat variabel "pesan" yang diinisialisasi dengan nilai 234. Ini merupakan pesan atau data yang akan ditandatangani.

Kemudian, terdapat perhitungan pada variabel "sign" yang menggunakan fungsi pow(). Fungsi ini menghitung hasil perpangkatan modular (modular exponentiation) dari pesan dengan eksponen dari kunci privat "kunci.d" dan modulus "kunci.n". Dalam konteks RSA, operasi perpangkatan modular ini digunakan untuk menghasilkan tanda tangan digital.

Hasil perpangkatan modular tersebut, yang merupakan tanda tangan digital, dicetak dalam representasi heksadesimal menggunakan fungsi "hex()" dan diikuti dengan teks "Digital Signature:".

Jadi, kode tersebut menghitung dan mencetak tanda tangan digital dalam bentuk heksadesimal dari pesan menggunakan kunci privat RSA yang disimpan dalam variabel "kunci".

### 3. Jalankan kode berikut

```
pesan = 234
verify = pow(sign, kunci.e, kunci.n)
print("Digital Signature valid:", pesan == verify)
```

## Penjelasan output



```
pesan = 234
verify = pow(sign, kunci.e, kunci.n)
print("Digital Signature valid:", pesan == verify)
```

[8] ✓ 0.0s

... Digital Signature valid: True

Muhammad Farisy Chaniago  
215150200111038

Pertama, terdapat variabel "pesan" yang diinisialisasi dengan nilai 234. Ini merupakan pesan asli yang ingin diverifikasi.

Selanjutnya, terdapat perhitungan pada variabel "verify" yang menggunakan fungsi pow(). Fungsi ini menghitung hasil perpangkatan modular (modular exponentiation) dari tanda tangan digital "sign" dengan eksponen dari kunci publik "kunci.e" dan modulus "kunci.n". Dalam konteks RSA, operasi perpangkatan modular ini digunakan untuk mendekripsi tanda tangan digital dan mendapatkan nilai yang akan dibandingkan dengan pesan asli.

Setelah itu, dilakukan perbandingan antara pesan asli "pesan" dan hasil dekripsi "verify". Jika pesan asli sama dengan hasil dekripsi, maka verifikasi dianggap berhasil dan hasilnya akan dicetak dalam bentuk teks "Digital Signature valid: True". Jika tidak, maka verifikasi dianggap gagal dan hasilnya akan dicetak dalam bentuk teks "Digital Signature valid: False".

Jadi, kode tersebut melakukan verifikasi terhadap tanda tangan digital dengan menggunakan kunci publik RSA yang disimpan dalam variabel "kunci". Hasilnya akan mencetak apakah tanda tangan digital valid atau tidak berdasarkan perbandingan dengan pesan asli.

#### 4. Jalankan kode berikut

```
pesan = "PesanRahasia"
sign = pow(pesan, kunci.d, kunci.n)
print("Digital Signature:", hex(sign))
```

## Penjelasan output



```
pesan = "PesanRahasia"
sign = pow(pesan, kunci.d, kunci.n)
print("Digital Signature:", hex(sign))
```

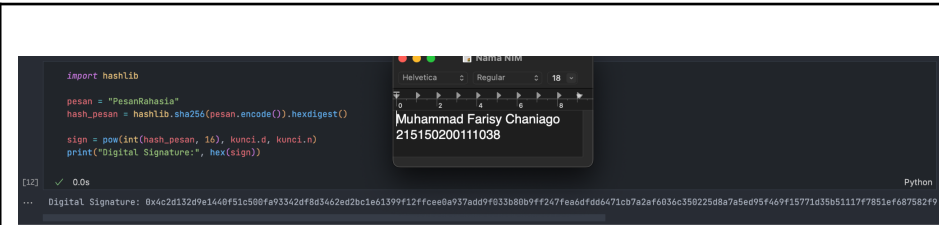
[11] 0.0s

Traceback (most recent call last):  
 File /Users/farisy/PraktikumKI/digital-sign.ipynb, Cell 4 in 2  
 1 pesan = "PesanRahasia"  
----> 2 sign = pow(pesan, kunci.d, kunci.n)  
 3 print("Digital Signature:", hex(sign))

TypeError: unsupported operand type(s) for \*\* or pow(): 'str', 'int', 'int'

Output yang dihasilkan dari kode adalah error, karena pesan yang ingin diubah bertipe String. Fungsi pow() tidak menerima jenis pesan String.

5. Koreksilah hasil error pada Nomor 4 dengan menggunakan rekomendasi hasil search engine yang biasa Anda gunakan. . Screen capture kode hasil koreksi dan hasilnya.



```
import hashlib

pesan = "PesanRahasia"
hash_pesan = hashlib.sha256(pesan.encode()).hexdigest()

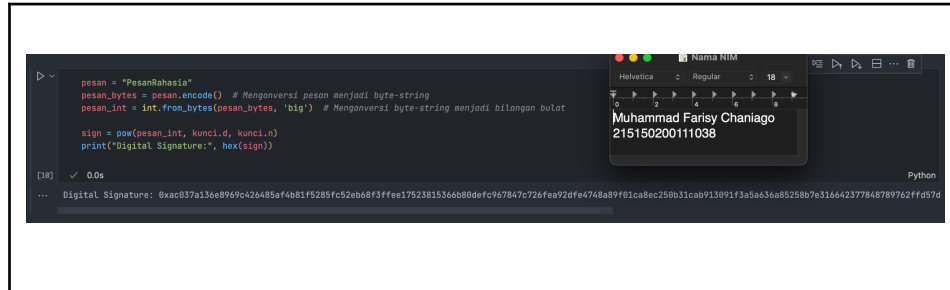
sign = pow(int(hash_pesan, 16), kunci.d, kunci.n)
print("Digital Signature:", hex(sign))
```

[12] 0.0s

Digital Signature: 0x4c2d132dfe1440f51c500fe93342df9d5462ed2bc1e61399f12ffce8e937add9f033b0b9ff247f6e6dfdd6471cb7a2af6036c350225d8a7a5ed95f469f15771d35b51117f7851ef687582f9

Python

6. Koreksilah hasil error pada Nomor 4 dengan menggunakan hasil rekomendasi ChatGPT. . Screen capture kode hasil koreksi dan hasilnya.



```
pesan = "PesanRahasia"
pesan_bytes = pesan.encode() # Mengonversi pesan menjadi byte-string
pesan_int = int.from_bytes(pesan_bytes, 'big') # Mengonversi byte-string menjadi bilangan bulat

sign = pow(pesan_int, kunci_d, kunci_n)
print("Digital Signature:", hex(sign))
```

[10] ✓ 0.0s ... Digital Signature: 0xac837a136e899c426485af4b81f5285fc52eb68f3ffee1752381536688defc967847c726fee92dfe4748a97f01ca8ec250b31cab713091f3a5a636a85258b7e316642377848789762ffd57d

Nama NIM  
Muhammad Farisy Chaniago  
215150200111038

Python

7. Amati hasil dari output langkah 5 dan 6, apakah yang dapat disimpulkan?

Terdapat perbedaan kode yang digunakan dari koreksi oleh ChatGPT dan google. Pada ChatGPT, pesan yang awalnya bertipe String diubah menjadi bytes lalu diubah lagi menjadi integer sehingga fungsi `pow()` bisa dijalankan. Sedangkan, solusi dari google menggunakan modul `hashlib` untuk mengubah pesan String menggunakan SHA256 lalu String heksadesimal diubah menjadi integer untuk dijalankan dengan fungsi `pow()`.

## 7.5. KESIMPULAN

Enkripsi digunakan untuk menjaga kerahasiaan data, dekripsi digunakan untuk mendekripsi data yang telah dienkripsi, hashing digunakan untuk verifikasi integritas data, dan tanda tangan digital digunakan untuk mengautentikasi dan memverifikasi asal-usul data. Ketiga konsep ini merupakan komponen penting dalam menciptakan keamanan dan kepercayaan dalam komunikasi dan penyimpanan data.

## 7.6. EVALUASI

1. Jelaskan perbedaan dari proses enkripsi dan hashing!
2. Jalankan algoritma hashing lain yaitu SHA384, dan amati hasil outputnya dan simpulkan.

1. Perbedaannya :

A. Tujuan Utama:

- Enkripsi: Tujuan utama dari proses enkripsi adalah menjaga kerahasiaan data dengan mengubahnya menjadi bentuk yang tidak dapat dibaca atau dimengerti. Enkripsi melibatkan penggunaan algoritma dan kunci enkripsi yang digunakan untuk mengubah data asli menjadi bentuk terenkripsi yang hanya dapat diakses oleh penerima yang memiliki kunci dekripsi yang benar.
- Hashing: Tujuan utama dari proses hashing adalah memverifikasi integritas data. Hashing menggunakan algoritma hash kriptografi untuk menghasilkan nilai hash yang unik untuk setiap data input. Nilai hash yang dihasilkan bergantung pada konten data input. Setiap perubahan kecil pada data input akan menghasilkan nilai hash yang berbeda. Hashing tidak dapat dikembalikan ke bentuk aslinya dan tidak digunakan untuk mengenkripsi atau mendekripsi data.

B. Reversibilitas:

- Enkripsi: Proses enkripsi dapat secara terbalik dibalik dengan menggunakan kunci dekripsi yang benar. Artinya, data yang telah dienkripsi dapat dikembalikan ke bentuk aslinya (didekripsi) oleh penerima yang memiliki kunci dekripsi yang sesuai.
- Hashing: Proses hashing tidak dapat secara terbalik dibalik. Nilai hash yang dihasilkan tidak dapat dikembalikan ke bentuk asli data. Ini berarti tidak ada mekanisme untuk mendapatkan data asli dari nilai hash.

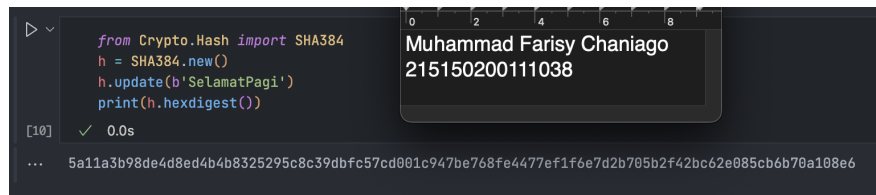
C. Kunci:

- Enkripsi: Proses enkripsi melibatkan penggunaan kunci enkripsi dan dekripsi. Data dienkripsi menggunakan kunci enkripsi dan hanya dapat didekripsi dengan kunci dekripsi yang sesuai.
- Hashing: Proses hashing tidak melibatkan penggunaan kunci. Algoritma hash menghasilkan nilai hash yang unik berdasarkan konten data input, tetapi tidak ada kunci yang digunakan untuk menghasilkan nilai hash atau mengubahnya.

D. Penggunaan:

- Enkripsi: Enkripsi digunakan untuk melindungi kerahasiaan data dan mencegah akses yang tidak sah. Hal ini umumnya digunakan dalam komunikasi aman, penyimpanan data yang sensitif, dan melindungi privasi pengguna.
- Hashing: Hashing digunakan untuk memverifikasi integritas data. Ini berguna dalam validasi integritas file, perbandingan data, penyimpanan kata sandi yang aman, dan keperluan lain di mana penting untuk memastikan bahwa data tidak berubah.



A screenshot of a Python IDE. The code editor shows the following Python code: 

```
from Crypto.Hash import SHA384
h = SHA384.new()
h.update(b'SelamatPagi')
print(h.hexdigest())
```

 Below the code, the output is displayed: 

```
[10] ✓ 0.0s
... 5a11a3b98de4d8ed4b4b8325295c8c39dbfc57cd001c947be768fe4477ef1f6e7d2b705b2f42bc62e085cb6b70a108e6
```

 A small window in the top right corner displays the name "Muhammad Farisy Chaniago" and the number "215150200111038".

- Pertama, lakukan impor kelas SHA384 dari modul `Crypto.Hash` menggunakan pernyataan “`from Crypto.Hash import SHA384`”. Ini memungkinkan untuk menggunakan algoritma SHA-384 untuk menghasilkan hash.
- Kemudian, buat objek hash baru dengan “`h = SHA384.new()`”. Ini menginisialisasi objek hash dengan algoritma SHA-384.
- Selanjutnya, perbarui objek hash dengan pesan menggunakan metode “`update(b'SelamatPagi')`”. Dalam kasus ini, pesan yang diupdate adalah byte-string “SelamatPagi”.
- Akhirnya, cetak hash yang dihasilkan menggunakan metode “`hexdigest()`” dalam bentuk heksadesimal dengan “`print(h.hexdigest())`”. Metode ini mengembalikan hash sebagai string heksadesimal yang mewakili nilai hash dari pesan.

Jadi, kode tersebut menghitung dan mencetak hash SHA-384 dari pesan “SelamatPagi” menggunakan modul `Crypto.Hash` dan algoritma SHA-384.