CSA NOV 2024 Assignment 1

Cryptography

Prepared By : S Divya

Table of Contents - Overview

1. Encode your name using Base64 and provide the resulting text. Decode it back to your name

2. Encrypt your name using AES + ECB and key hackerspace12345 and provide the cipher text - Try to decrypt it back

3. Generate an RSA Key pair - Use public key encrypt your name and provide the resulting ciphertext - try to decrypt the ciphertext using the private key

4. Generate SHA256 hash of your name and provide the resulting hash - Try to reverse the hashing by any method

1. In Linux Command line, Encode your name using Base64 and provide the resulting text. Decode it back to your name

Encoding the Name

The string "S Divya" is to be encoded into a Base64 string because Base64 is used to represent binary data in an ASCII string format. Each Base64-encoded character represents 6 bits of the original binary data.

Text to Encode: **"S Divya"**

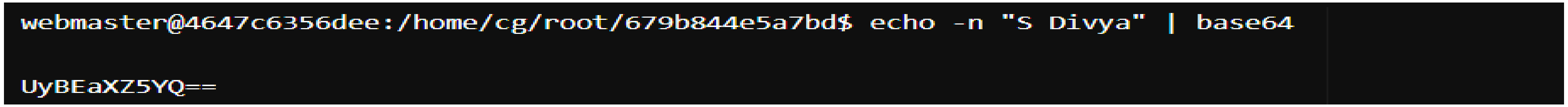
Encoding "S Divya" using Base64

To encode your name "S Divya" in Base64 on the Linux command line, use the following command:

$echo -n "S Divya" | base64

On running the above command, will get the following result: **UyBEaXZ5YQ==**

Output



```
webmaster@4647c6356dee:/home/cg/root/679b844e5a7bd$ echo -n "S Divya" | base64

UyBEaXZ5YQ==
```

Decoding the Base64 String

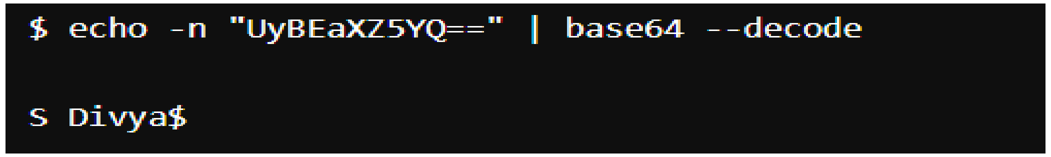
On decoding the Base64 string, the base64 -- decode command reverses this process and recovers the original text ("S Divya").

To decode the encoded string back into its original form do this, run the following command:
$echo -n "UyBEaXYhYQ==" | base64 --decode

base64 --decode: This tells the base 64 command to decode the string from Base64 back to the original text.

Output

On running this command in the terminal will output: **S Divya**



```
$ echo -n "UyBEaXZ5YQ==" | base64 --decode

S Divya$
```

Decoded result: When decoded **"UyBEaXZ5YQ=="** gives the original string **"S Divya“** as originally encoded
So, The result after decoding **"UyBEaXZ5YQ=="** is **"S Divya“**

•**Original Text** : "S Divya"
•**Base64 encoded result**: "UyBEaXZ5YQ=="
•**Decoded result**: "S Divya"

2. In Linux Command line, Encrypt your name using AES + ECB and key hackerspace12345 and provide the cipher text - Try to decrypt it back

AES Encryption and Decryption with ECB Mode

AES requires a key of a specific length. In the case given, the key **"hackerspace12345"** is already 16 characters long, which corresponds to 128 bits (1 byte = 8 bits, so 16 bytes = 128 bits).

Encrypting "S Divya" using AES-128-ECB

Open a terminal and use the openssl command to encrypt the string "S Divya" with the key **hackerspace12345**

For this first convert the key hackerspace12345 to hexadecimal:

```
$echo -n "hackerspace12345" | xxd –p
```

This will give the hexadecimal representation of the string hackerspace12345.

The output will be: 6861636b65727370616365313233345

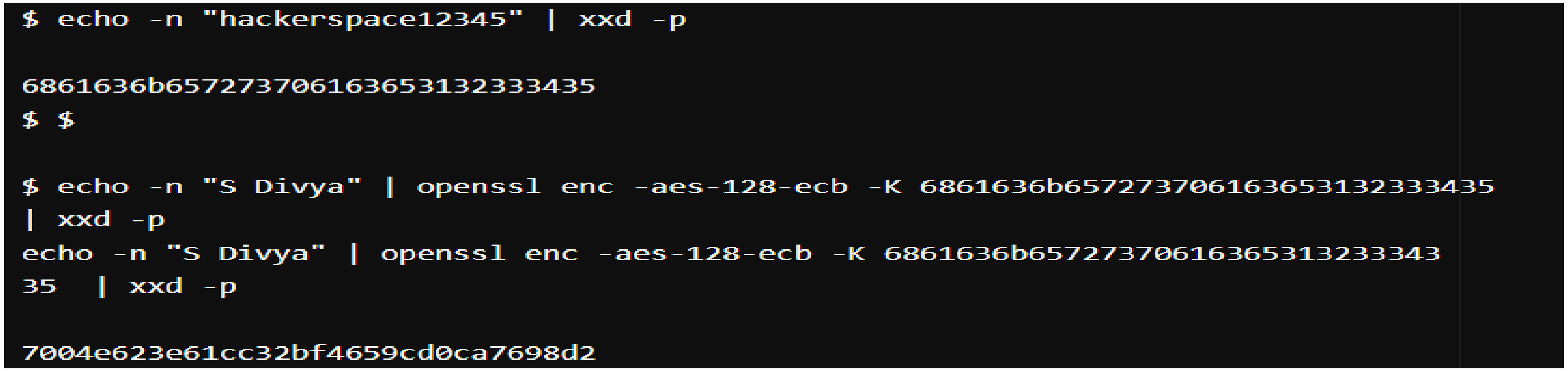
Using the correct hex key to encrypt the name "S Divya". Now run:

```
$ echo -n "S Divya" | openssl enc -aes-128-ecb -K 6861636b65727370616365313233345 | xxd -p
```

- **echo -n “S Divya” :** Outputs the string "S Divya" without a newline
- **openssl enc -aes-128-ecb :** Specifies the encryption method: AES with a 128-bit key and ECB (Electronic Codebook) mode.
- **-K 6861636b657273706163653132333435 :** The encryption key, which must be specified in hexadecimal.
- **xxd -p :** Converts the output into a pure hexadecimal format for better readability.

Output

The AES-128-ECB encrypted version of "S Divya“ :
This will output the encrypted cipher text in hexadecimal format : **7004e623e61cc32bf4659cd0ca7698d2**



```
$ echo -n "hackerspace12345" | xxd -p

6861636b657273706163653132333435
$ $

$ echo -n "S Divya" | openssl enc -aes-128-ecb -K 6861636b657273706163653132333435
| xxd -p
echo -n "S Divya" | openssl enc -aes-128-ecb -K 6861636b6572737061636531323334З
35  | xxd -p

7004e623e61cc32bf4659cd0ca7698d2
```

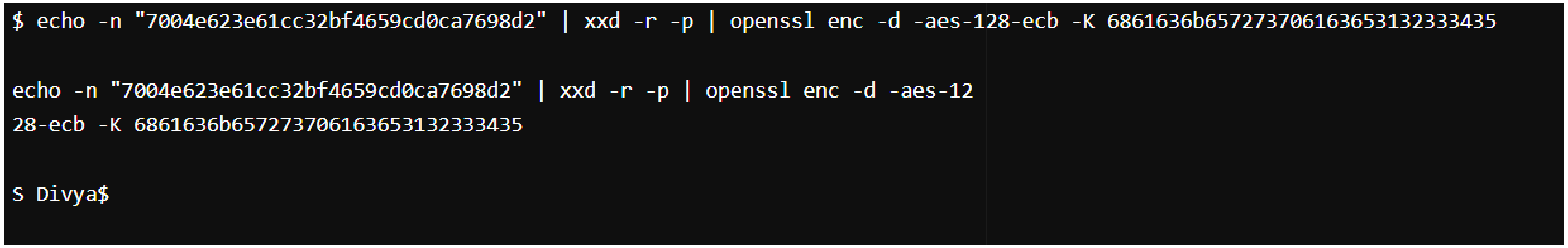
Decrypting the Cipher Text Back to "S Divya“

To decrypt the cipher text back into the original name, use the following openssl command :
$echo -n " 7004e623e61cc32bf4659cd0ca7698d2 " | xxd -r -p | openssl enc -d -aes-128-ecb -K 6861636b657273706163653132333435

- **echo -n " 7004e623e61cc32bf4659cd0ca7698d2 " :** This is the cipher text that is to be decrypted
- **xxd -r –p :** Converts the hex string back to binary format
- **openssl enc -d -aes-128-ecb :** Specifies that wanted to decrypt using AES-128 in ECB mode.

Output

The output of decryption : **S Divya**



```
$ echo -n "7004e623e61cc32bf4659cd0ca7698d2" | xxd -r -p | openssl enc -d -aes-128-ecb -K 6861636b657273706163653132333435

echo -n "7004e623e61cc32bf4659cd0ca7698d2" | xxd -r -p | openssl enc -d -aes-12
28-ecb -K 6861636b657273706163653132333435

S Divya$
```

Ciphertext (in hex): 7004e623e61cc32bf4659cd0ca7698d2
Decrypted plaintext: S Divya

3. In the Linux Command line, Generate an RSA Key pair - Use public key encrypt your name and provide the resulting ciphertext - try to decrypt the ciphertext using the private key

Generating an RSA Key Pair

Generate two keys: the **private key** (Used for decryption) and the **public key** (Used for encryption). These keys are mathematically linked.

In RSA, encryption can only be performed with the public key, while decryption can only happen with the private key.

Encryption using the Public Key

Using the **public key** to encrypt the message "**S Divya**". The encryption process transforms your plaintext into ciphertext (scrambled data) which cannot be read by anyone without the private key.

Decryption using the Private Key

The encrypted message (ciphertext) can only be decrypted by someone with the **private key**. By using the private key to decrypt, should get back the original message, which is **"S Divya"** in this case.

Generate an RSA Key Pair using the openssl command

Run the following command to generate the key pair: **$openssl genpkey -algorithm RSA -out private_key.pem**
This will generate the private key file private_key.pem

Next, extract the **public key** from the private key: **$ openssl rsa -pubout -in private_key.pem -out public_key.pem**
This creates the public key file public_key.pem which will be used for encryption.

Encrypt Your Name with the Public Key

Use the **public key** to encrypt name **"S Divya“** for this use the following command :

$ echo -n "S Divya" | openssl rsautl -encrypt -inkey public_key.pem -pubin -out encrypted_name.bin

- **echo -n "S Divya" :** This will output the name "S Divya“
- **openssl rsautl -encrypt :** This uses RSA encryption
- **-inkey public_key.pem :** Specifies the public key file
- **-pubin :** Tells OpenSSL that the key provided is a public key
- **-out encrypted_name.bin :** Specifies the output file where the encrypted ciphertext will be saved

After running this command, you will have the encrypted data stored in the **encrypted_name.bin** file

To view the ciphertext in hexadecimal form, run : **$ xxd -p encrypted_name.bin**

Output:

The string of hexadecimal characters which is the encrypted form of “S Divya” is :

```
4126b63abdc9ac23a44843d318dec4b3840c83b7cd154f1ea4c8661bb59562b36db47193148b74f9152fae29b075a4c6e9d4dc78
7355769ac77ce4fdb7c989a3503624b3e924f6cc0cc2c86df2f82939bf7fcc1e00391f3b7bec2a8c23766eb7903422ce5518a64899
422fc7009a949614ba7fb554c1ed9d5037374384b665630b59075b63969c7e72dd61a59e6531d5e4d1a696d153
1e2187f21073a578814df60f723fe3976590e8f14f3c1797bf8b3bf6e8c32659f16ef62df30a94f2a32991831a5da2734da9bdfeb9f
870c7b8b8b88e4d08e46cbe2f29bd30ad35a0e57fdf74f06954192d660d8580b3f5c5c8d6ccfce3477f3947946319607849545b6a
```

Decrypt the Ciphertext with the Private Key

The ciphertext is decrypted using the private key.

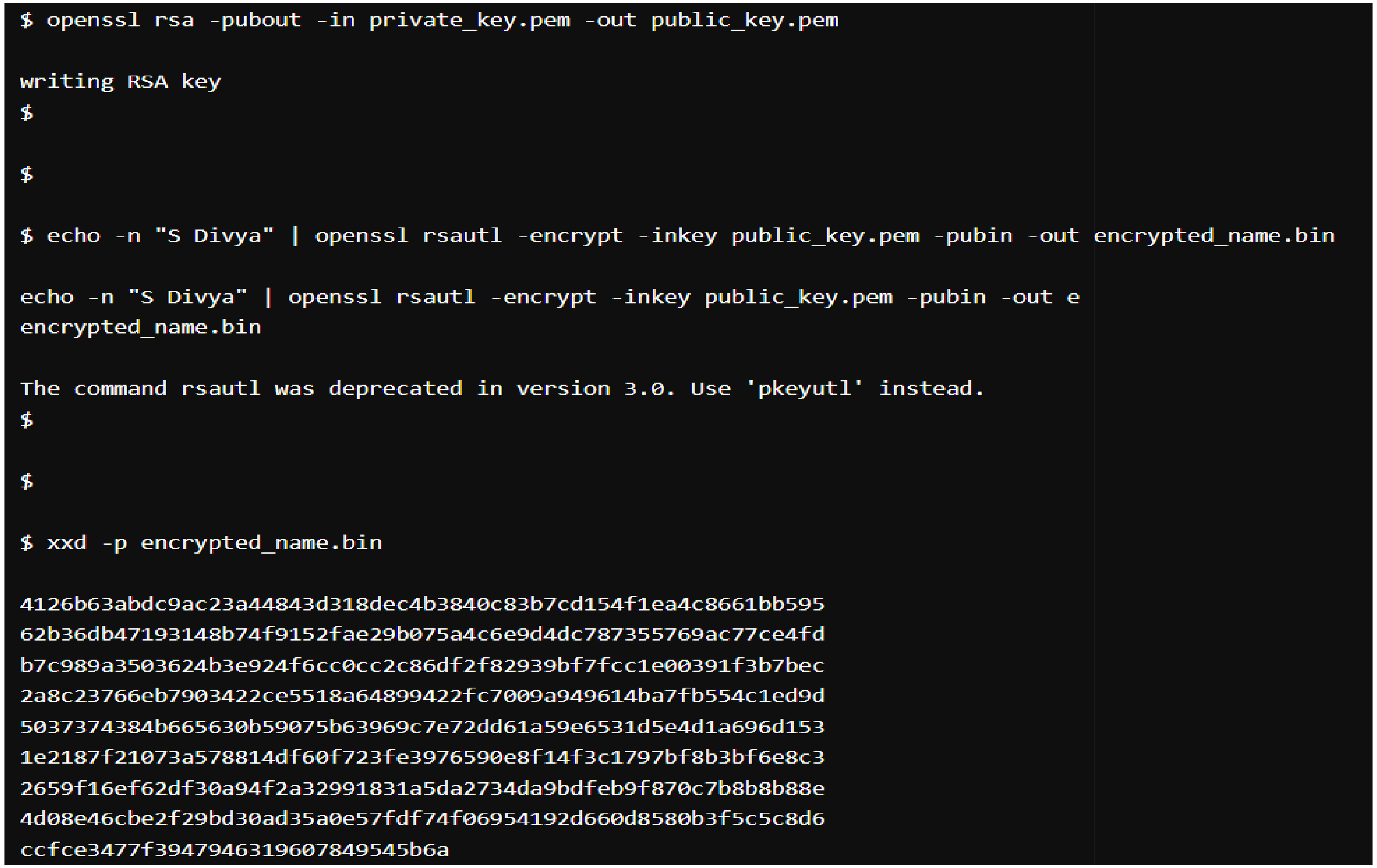
Run the following command to decrypt the ciphertext:

```
$openssl rsautl -decrypt -inkey private_key.pem -in encrypted_name.bin
```

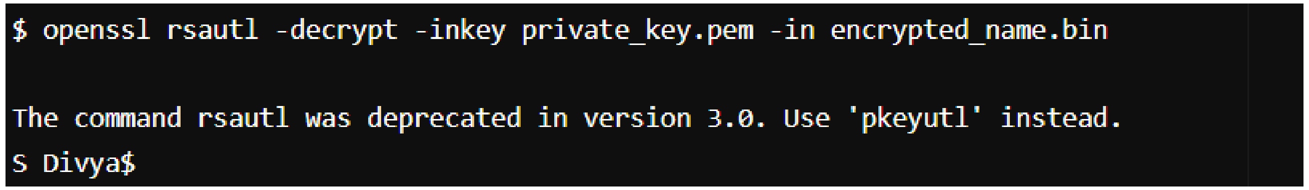
Output:

This will output (Decrypted message) : **S Divya**

Output



```
$ openssl rsa -pubout -in private_key.pem -out public_key.pem

writing RSA key
$

$

$ echo -n "S Divya" | openssl rsautl -encrypt -inkey public_key.pem -pubin -out encrypted_name.bin

echo -n "S Divya" | openssl rsautl -encrypt -inkey public_key.pem -pubin -out e
encrypted_name.bin

The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.
$

$

$ xxd -p encrypted_name.bin

4126b63abdc9ac23a44843d318dec4b3840c83b7cd154f1ea4c8661bb595
62b36db47193148b74f9152fae29b075a4c6e9d4dc787355769ac77ce4fd
b7c989a3503624b3e924f6cc0cc2c86df2f82939bf7fcc1e00391f3b7bec
2a8c23766eb7903422ce5518a64899422fc7009a949614ba7fb554c1ed9d
5037374384b665630b59075b63969c7e72dd61a59e6531d5e4d1a696d153
1e2187f21073a578814df60f723fe3976590e8f14f3c1797bf8b3bf6e8c3
2659f16ef62df30a94f2a32991831a5da2734da9bdfeb9f870c7b8b8b88e
4d08e46cbe2f29bd30ad35a0e57fdf74f06954192d660d8580b3f5c5c8d6
ccfce3477f3947946319607849545b6a
```



```
$ openssl rsautl -decrypt -inkey private_key.pem -in encrypted_name.bin

The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.
S Divya$
```

4. In the Linux Command line, Generate SHA256 hash of your name and provide the resulting hash - Try to reverse the hashing by any method

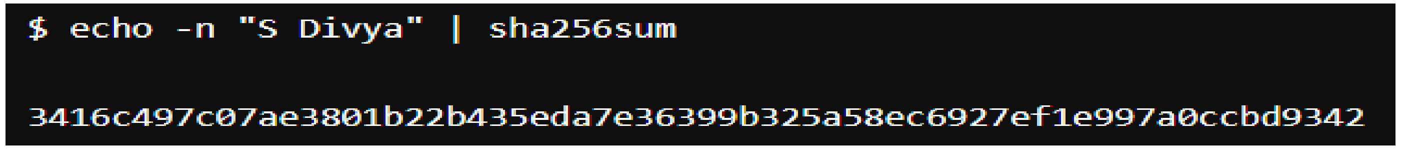
<u>Generating SHA256 Hash for "S Divya“</u>

Use the sha256sum command to generate the SHA-256 hash. For this run the following command :

$ echo -n "S Divya" | sha256sum

- **echo -n "S Divya" :** This will output your name "S Divya" without a newline
- **sha256sum :** This command calculates the SHA-256 hash of the input

<u>Output:</u>



```
$ echo -n "S Divya" | sha256sum

3416c497c07ae3801b22b435eda7e36399b325a58ec6927ef1e997a0ccbd9342
```

SHA256 Hash of 'S Divya’:

3416c497c07ae3801b22b435eda7e36399b325a58ec6927ef1e997a0ccbd9342

Reversing SHA256 Hash

SHA256 is a **one-way cryptographic function**, meaning that reversing it directly (or finding the original input from the hash) is computationally infeasible.

Why it's hard to reverse:

One-Way Nature of Hash Functions

- SHA256 is a **one-way function,** meaning that once applying the algorithm to the input, it's designed to be irreversible. So that it's almost impossible to retrieve the original input from the hash.
- Once you hash the name "S Divya" the resulting hash does not contain any inherent reversible relationship with the original input.
- **You can easily generate a hash from an input**, but there's no straightforward way to take the hash and regenerate the input data.

Pre-image Resistance

•This property means that even knowing the output (the hash value), it's nearly impossible to reverse-engineer the original input. Without the original input (in this case, the name "S Divya"), there's no way to directly get back to it.

•A SHA256 hash doesn't store any information that could directly point back to the original string. It only generates a fixed-length output that doesn't map back easily to the input.

Brute-Force Attack

•Could attempt to use a **brute-force attack**, where to try every possible combination of strings, hash each one, and check if it matches the original hash. However, given the vast number of possible inputs, this would take an impractical amount of time.

Rainbow tables

•They are precomputed tables of hash values for common inputs, but they don't typically include names like "S Divya" Additionally, this approach only works for simple, predictable inputs.

Conclusion: Reversing SHA256

- Reversing a SHA256 hash is **not feasible**. There's no practical way to reverse a SHA256 hash directly back to its original string (which in the case is my name)
- Therefore, once generated the hash of **"S Divya"** using SHA256, there's **no efficient way to reverse it** back to the original string without already knowing what the string was.
- The only reliable way to know what the original input was is if you already have it, or if you happen to encounter it in a precomputed hash table (which is unlikely for unique names).
- In short, the hash **cannot be reversed**, which is a critical property of cryptographic hashes like SHA256. This is what makes SHA256 and other cryptographic hash functions so secure.