CYBER SECURITY AND ETHICAL HACKING

REVANTH P
20BPS1161
VIT mail id – penta.revanth2020@vitstudent.ac.in

ASSESSMENT -3

Cryptography Analysis and Implementation

AIM: To analyze cryptographic algorithms and implement them in a practical scenario.

Cryptographic Algorithm:
It is sequences of processes, or rules, used to encipher and decipher messages in a cryptographic system. In other words, they're processes that protect data by making sure that unwanted people can't access it. These algorithms have a wide variety of uses, including ensuring secure and authenticated financial transactions.

There are 3 types of cryptographic algorithm:

1. **Symmetric key algorithm:** A symmetric-key algorithm transforms data to make it extremely difficult to view without possessing a secret key. The key is considered symmetric because it is used for both encrypting and decrypting. These keys are usually known by one or more authorized entities.
2. **Asymmetric-key algorithm:** An asymmetric-key algorithms use paired keys (a public and a private key) in performing their function. The public key is known to all, but the private key is controlled solely by the owner of that key pair. The private key cannot be mathematically calculated through the use of the public key even though they are cryptographically related.
3. **Hash Functions:** There is no usage of any key in this algorithm. A hash value with fixed length is calculated as per the plain text which makes it impossible for contents of plain text to be recovered. Many operating systems use hash functions to encrypt passwords.

Analysis:
1. Symmetric Cryptographic Algorithm (CAESAR CIPHER):

Brief Explanation:

This cypher is a sort of substitution cypher in which each letter in the plaintext is replaced by a letter that is located a fixed number of positions further down the alphabet. A left shift of three, for instance, would result in D being replaced by A, E becoming B, and so on.

Key advantages and strengths:

One of the easiest encryption methods is the Caesar cypher. To receive the cypher text, it entails shifting each letter of the plaintext by a predetermined number of places. The method is simple, which makes it easy for even newbies to understand and use.

Speed: The Caesar cypher makes encryption and decryption quick and easy. Shifting each letter in plaintext is a simple task that can be done quickly. This makes it a good choice for situations where speed is important.

For learning purposes, the Caesar cypher is often used as an example of how to start learning about cryptography. Its ease helps people who are just starting out understand basic ideas like encryption, decryption, and the idea of a key.

Known Weaknesses or Vulnerabilities:

- Lack of Encryption Complexity: The Caesar cypher uses a simple replacement method in which each letter is moved by a set amount. Due to its lack of comp lexity, it is easy to break using methods like statistical analysis, pattern recognition, or known textual attacks.
- Dependence on language: The Caesar cypher was made for the English code. If you use it with languages that use different alphabets or letter sets, it might not work or give you useful results.
- Not Good Enough for Modern Cryptographic Needs: The Caesar cypher is thought to be too old and not safe enough for modern cryptographic needs. Because of its flaws, it can't be used to protect private information or ensure secure contact, which requires stronger encryption algorithms and methods.

Real-World Examples:

The Caesar cypher can be used to send easy messages with a basic level of encryption between friends or family members. Also, It is often used to teach the basics of cryptography in training or workshops for beginners. Students can use the cypher to practice encrypting and decrypting messages to learn about replacement cyphers, key spaces, and the basic rules of encryption.

2. Asymmetric Cryptographic Algorithm (RSA):

Brief Explanation:

The algorithm is based on how hard it is to factor big numbers. It includes coming up with a set of mathematically related keys: a public key for encryption and a private key for decryption. The security of RSA depends on how hard it is to break down big numbers into their prime factors.

Key advantages and strengths:

- Key sharing: RSA allows secure key sharing. With RSA, the public key mn be given to anyone, while the private key stays secret. This makes it easier for people who have never shared a secret key to communicate securely and makes it easier to handle encryption keys.
- Digital Signatures: RSA is widely used to create and check digital signatures. Digital signatures ensure the integrity and validity of digital papers or messages. The private key is used to create a digital signature, which can be checked by anyone with the matching public key. This makes sure that the signed data is correct and cannot be changed.

- Secure Key Exchange: RSA can be used to do secure key exchange in asymmetric key systems. By encrypting a randomly produced symmetric key with the recipient's public key, RSA allows for secure key exchange, which can then be used for symmetric encryption. Without the need to send symmetric keys directly, this enables effective and secure contact between parties.

Known Weaknesses or Vulnerabilities:

- Key Management and Distribution: For RSA, encryption keys need to be carefully managed. Key generation, storage, and sharing can be difficult, especially when there are a lot of people. It can be hard to keep private keys secure and make sure they get to the right people, and ifa private key is lost or stolen, it can result in a full security breach.
- When encrypting or decrypting big amounts of data, RSA is slower than symmetric encryption methods. As a result, it is frequently used in conjunction with symmetric encryption methods for encrypting real data while RSA is used for key sharing or digital signatures.
- Vulnerability to Attacks: RSA is vulnerable to attacks like timing attacks, side-channel attacks, and picked ciphertext attacks. These attacks take advantage of bugs in the way the algorithm is implemented or flaws in the algorithm itself. To reduce these risks, you need countermeasures like random padding schemes and secure application practices.

Real-World Examples:

RSA is utilized in a broad variety of applications, including secure email (PGP), SSL/TLS certificates for website authentication, and secure communication protocols such as SSH.

3. Hash Function: Secure Hash Cryptographic Algorithm (SHA-256):

Brief Explanation:

SHA-512 (Secure Hash Algorithm 512-bit) is a secure hash code that is part of the SHA-2 family. It works with 64-bit blocks and makes a hash value with 512 bits. SHA-512 uses a series of logical operations, bitwise operations, and moduhr arithmetic to process the raw data and make a fixed-size hash.

Key advantages and strengths:

- SHA-512's security is better than SHA-256's. The larger hash size (512 bits) gives a larger output space, which makes it impossible to find clashes or figure out the original input from the hash value.
- Brute-Force Attack Resistance: SHA-512's bigger hash size improves its resistance to brute-force attacks. With a 512-bit hash value, the number of possible hash outputs is much higher.

This makes it impossible to guess the original input through extensive searching or trial-and-error methods.

- SHA-512 ensures message integrity by generating a fixed-size hash number for any input. Even a small change in the data input will result in a totally different hash output. This makes the hash very sensitive to data changes and gives a strong method for checking the integrity of the data.

Known Weaknesses or Vulnerabilities:

- Computing Overhead: Since SHA-512 has a bigger hash size (512 bits), it needs more computing resources than hash functions with smaller output sizes. This can slow things down a little, especially when handling a lot of data or doing a lot of hash tasks.
- Memory Use: SHA-512 uses 64-bit blocks, which can use more memory than hash functions that use smaller blocks. This can be important in places with limited resources or on devices with limited memory.
- Digest Length: The larger output size of SHA-512 (512 bits) may not always be necessary or possible for some uses. When a smaller hash number is enough, using SHA-512 can lead to extra storing needs and work

Implementation:

Caeser Cipher Algorithm(Python)

Code:

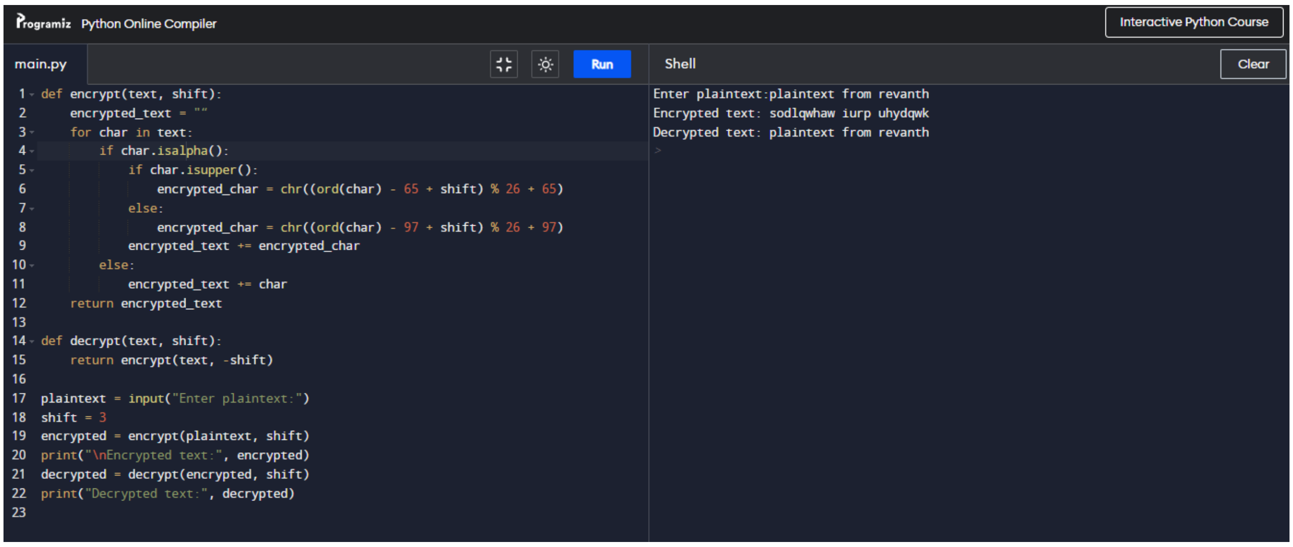
```
def encrypt(text, shift): encrypted_text = "“ for char in text:
if char.isa1pha():
if char.isupper():
encrypted_char = chr((ord(char) - 65 + shift) % 26 + 65) else:
encrypted_char = chr((ord(char) - 97 + shift) % 26 + 97) encrypted_text += encrypted_char
else:
encrypted_text += char return encrypted_text
def decrypt(text, shift): return encrypt(text, -shift)
plaintext = input(“Enter plaintext: “)
shift = input(“Enter the ’number’ of shifts: “)
encrypted = encrypt(plaintext, shift)
print("\nEncrypted text:", encrypted)
decrypted = decrypt(encrypted, shift)
```

print("Decrypted text:", decrypted)

Explanation:

- The plaintext is moved down the alphabet to encrypt it.
- We have *encrypt)* and *decryptJ) vn* this code. A plaintext message and a shift value are the inputs of the *encrypt()* function. The message is encrypted by moving each letter by the shift value.
- The plaintext is checked to determine the case. Ifthe characters are uppercase, we convert them to their ASCII value, subtract 65 (the ASCII value of'A'), add the shift value, take the modulus of26 to wrap around the alphabets, add 65 again, and convert it back to characters.
- Lowercase letters are calculated by subtracting 97 (the ASCII value of 'a') instead of 65. This makes sure uppercase and lowercase letters are shifted appropriately.
- Finally, *decrypt()* is essentially *encrypt()* with a negative shift value. Thus, we may decrypt a message encrypted with a shift of 3 by using -3.

Screenshot:



```python
def encrypt(text, shift):
    encrypted_text = ""
    for char in text:
        if char.isalpha():
            if char.isupper():
                encrypted_char = chr((ord(char) - 65 + shift) % 26 + 65)
            else:
                encrypted_char = chr((ord(char) - 97 + shift) % 26 + 97)
            encrypted_text += encrypted_char
        else:
            encrypted_text += char
    return encrypted_text

def decrypt(text, shift):
    return encrypt(text, -shift)

plaintext = input("Enter plaintext:")
shift = 3
encrypted = encrypt(plaintext, shift)
print("\nEncrypted text:", encrypted)
decrypted = decrypt(encrypted, shift)
print("Decrypted text:", decrypted)
```

```
Enter plaintext:plaintext from revanth
Encrypted text: sodlqwhaw iurp uhydqwk
Decrypted text: plaintext from revanth
>
```

Security Analysis:

Identify potential threats or vulnerabilities that could be exploited:

Input checking is lacking, which could be a vulnerability. The input data must be acceptable and free of any special characters or numbers in order for the code to function. Unexpected behaviour or even code execution flaws could result from bad input from an attacker, such as symbols or numbers.

- The fixed shift number is another vulnerability. Using a fixed shift of3 is easy for enemies who know the Caesar Cypher to figure out. A more secure method would be to use a random or variable shift number that changes with each encryption.

Propose countermeasures or best practices to enhance the security of your implementation:

- It’d be better if input validation is used, to make sure that only proper alphabetic characters are allowed as input and to handle any other cases correctly.

- We can set up a safe key management system to protect the encryption key used in the encryption and decoding process. This could include using encryption key boxes or hardware security modules (HSMs).

Discuss any limitations or trade-offs encountered during the implementation process:

- The Caesar Cypher is limited by how simple it is. It can be attacked by brute force, in which an attacker tries all of the shift numbers until the right one is found.

- The set shift value limits the number of keys, which makes it vulnerable to frequency analysis attacks. In these attacks, an attacker looks at how often letters appear in the ciphertext to figure out the most likely shift value.

- The Caesar Cypher also doesn't protect characters that aren't from the alphabet. The ciphertext keeps non-alphanumeric characters from the plaintext, which an attacker could use to figure out trends or other information.

Conclusion:

Importance of cyber security and ethical hacking:

Cyber security methods stop viruses, ransomware, phishing, social engineering, and illegal network breaks. Organizations can reduce cyber threats by finding vulnerabilities and putting in place security policies.

- Privacy: With more digital links, privacy and safety is more important than ever. Cybersecurity protects personal data and data keeping, which protects privacy.

- Ethical Hacking for Vulnerability Assessment: White-hat hacking, penetration testing, or ethical hacking are all terms used to describe authorized hacking to find system and network vulnerabilities. Ethical hackers help companies find security vulnerabilities and fix them before criminals do.

- Compliance with Legal and Regulatory Requirements: Banking, healthcare, and the government all have laws and rules about data safety. Following best practices for cyber security and doing frequent security checks and reviews help ensure compliance.

- Business Continuity and Reputation Management: Successful cyber strikes can cost companies money, stop them from doing their jobs, and hurt their names. Strong cybersecurity and danger tracking keep businesses going and protect their identities.

References:

1. Gowda, Shreyank N. "Innovative enhancement of the Caesar cipher algorithm for cryptography." *2016 2nd International Conference on Advances in Computing, Communication, & Automatics (ICACCA)(Fall)*. IEEE, 2016.

2. Zhou, Xin, and Xiaofei Tang. "Research and implementation of RSA algorithm for encryption ard decryption." *Proceedings of 2011 6th international fomm on strategic technology*. Vol. 2. IEEE, 2011.
3. Sumagita, Meiliana, et al. "Analysis of secure hash algorithm (SHA) 512 for encryption process on wab based application." *International Journal of Cyber-Security and Digital Forensics (IJCSDF)* 7.4 (2018): 373-381.