University of Taipei

Computer Science

Homework Ch.6

Student ID: U10916024

Student: Cheng-Hao, Zhang

張呈顥

2023

1. Symmetric Encryption vs. Asymmetric Encryption

Symmetric and asymmetric encryption are two fundamental methods used in cryptography to secure data and communications.

I. Symmetric Encryption:

Symmetric encryption, also known as secret-key encryption, employs a single secret key for both encryption and decryption processes. The same key is used by both the sender and the recipient to encrypt and decrypt the data. It is called "symmetric" because the encryption and decryption algorithms are symmetric or identical. The key must be securely shared between the sender and receiver before communication can take place.

- Advantages of symmetric encryption:
  - Faster and more efficient compared to asymmetric encryption.
  - Well-suited for encrypting large amounts of data.
  - It can be implemented in hardware, making it suitable for resource-constrained devices.
- Disadvantages of symmetric encryption:
  - The primary concern is the secure distribution of the secret key to all intended parties.
  - Each pair of communicating parties needs a unique secret key, which can be challenging to manage in large-scale systems.

    - Offers no inherent mechanism for authentication or non-repudiation.

Common symmetric encryption algorithms include AES (Advanced Encryption Standard) and DES (Data Encryption Standard).

II. Asymmetric Encryption:

Asymmetric encryption, also known as public-key encryption, employs a pair of mathematically related keys: a public key and a private key. The public key is used for encryption, while the private key is used for decryption. Unlike symmetric encryption, the keys used for encryption and decryption are not the same. Each individual has their own unique key pair.

- Advantages of asymmetric encryption:
    - Offers a solution to the key distribution problem in symmetric encryption as the public keys can be freely distributed.
    - Enables secure communication even if the public key is intercepted.
    - Provides mechanisms for authentication and digital signatures, allowing for non-repudiation.
- Disadvantages of asymmetric encryption:
    - Slower and computationally more intensive than symmetric encryption.
    - Limited in terms of the amount of data that can be encrypted using asymmetric encryption.
    - Requires more computational resources, which can be a challenge for resource-constrained devices.

Common asymmetric encryption algorithms include RSA (Rivest-Shamir-Adleman) and ECC (Elliptic Curve Cryptography)

In practice, a combination of both symmetric and asymmetric encryption is often used. For instance, symmetric encryption is commonly used to secure the actual data, while asymmetric encryption is used for key exchange and authentication purposes. This hybrid approach combines the speed and efficiency of symmetric encryption with the secure key distribution provided by asymmetric encryption.

2. Digital signature

Digital signatures have various applications in different domains to ensure the authenticity, integrity, and non-repudiation of digital documents and transactions. Some common applications of digital signatures include:

I. Document Integrity and Authentication

Digital signatures can be used to verify the integrity and authenticity of digital documents. By applying a digital signature to a document, the sender can ensure that the document has not been tampered with during transit and that it originated from the claimed sender. This is crucial in scenarios such as contracts, legal documents, financial transactions, and sensitive communications.

II. Email Security

Digital signatures can be used to secure email communications. By digitally signing an email, the sender can provide assurance to the recipient that the email has not been modified and that it indeed came from the claimed sender. It helps prevent email spoofing and protects against tampering or unauthorized modifications.

III. Software and Firmware Updates

Digital signatures are often used to verify the authenticity and integrity of software and firmware updates. By digitally signing the updates, software vendors can ensure that the updates have not been tampered with and that they originate from a trusted source. This helps prevent the installation of malicious or unauthorized updates.

IV. Secure Online Transactions

Digital signatures play a crucial role in secure online transactions, such as e-commerce and online banking. They provide a mechanism for verifying the identity of the parties involved and ensuring the integrity of the transaction. Digital signatures can be used to sign financial transactions, contracts, and other important documents in online transactions.

V. Code and Software Distribution

Digital signatures are used to ensure the authenticity and integrity of software code and applications during distribution. Software developers can digitally sign their code to guarantee that it has not been modified or tampered with since the time of signing. This helps users verify the authenticity of the software and ensures that they are not installing malicious or compromised code.

VI. Government and Legal Applications

Digital signatures are widely used in government and legal applications. They enable the secure and legally recognized signing of electronic documents, such as tax forms, permits, licenses, and contracts. Digital signatures provide the same legal validity as handwritten signatures, making electronic transactions more efficient and secure.

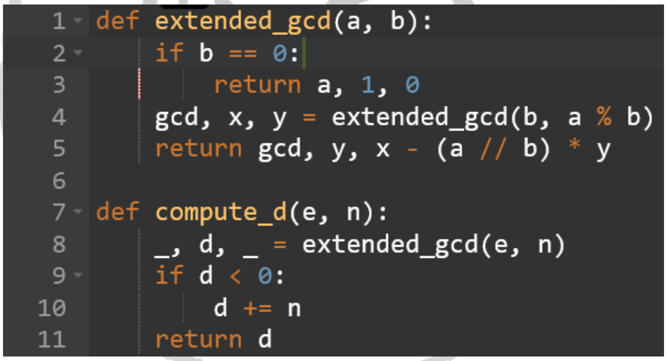
Overall, digital signatures provide a way to ensure the integrity, authenticity, and non-repudiation of digital documents, transactions, and communications, making them a crucial component of secure digital interactions in various domains.

3. RSA

$$p = 47, q = 71, and\ (p - 1)(q - 1) = 3220$$

$$find\ a, a \times 1019 = 1\ mod\ 3320$$

$$= a^{-1} \equiv 1019\ (mod\ 3320)$$



```python
def extended_gcd(a, b):
    if b == 0:
        return a, 1, 0
    gcd, x, y = extended_gcd(b, a % b)
    return gcd, y, x - (a // b) * y

def compute_d(e, n):
    _, d, _ = extended_gcd(e, n)
    if d < 0:
        d += n
    return d
```

Using compute_d(1019, 3220) = 79, a = 79

4. RSA

$M = 20, p = 3, q = 11$

$N = 33, \varphi(N) = (p-1)(q-1) = 20$

$choose\ a\ number\ e = 3\ coprimes\ \ \varphi(N)$

$and\ e \times d \equiv 1\ \big(mod\ \varphi(N)\big), d = 7$

Encryption: $C = M^e$ mod N $= 20^3$ mod $33 \equiv 14$

Decryption: $M = C^d$ mod N $= 14^7$ mod $33 \equiv 20$