Securing Data: Asymmetric Encryption and Digital Signature Implementation

Abidzhanov Ilias
Aitkozha Arshat
SE-2219

Introduction

This presentation will cover the implementation of **asymmetric encryption** and *digital signature* for securing data. We will explore the concepts, benefits, and practical applications of these techniques.

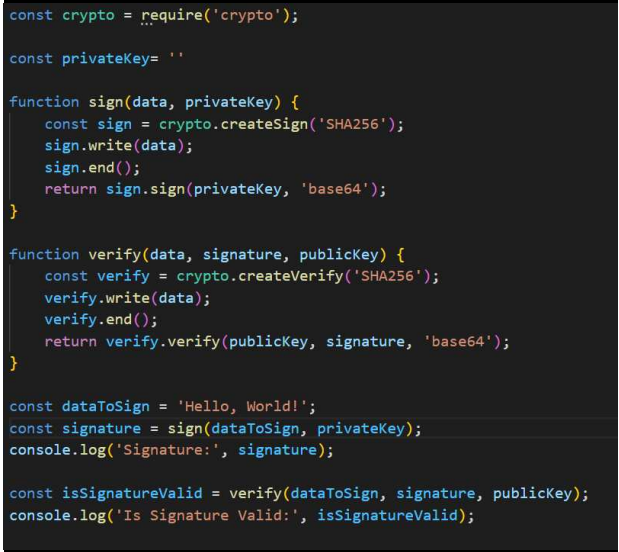
Asymmetric Encryption

Asymmetric encryption uses a pair of keys, public and private, for secure data transmission. The public key is used to encrypt data, while the private key is used to decrypt it. This method ensures secure communication over untrusted networks.

```
const crypto = require('crypto');

const { publicKey, privateKey } = crypto.generateKeyPairSync('rsa', {
  modulusLength: 2048,
  publicKeyEncoding: { type: 'spki', format: 'pem' },
  privateKeyEncoding: { type: 'pkcs8', format: 'pem' },
});

function encrypt(data, publicKey) {
    return crypto.publicEncrypt(publicKey, Buffer.from(data)).toString('base64');
}

function decrypt(encryptedData, privateKey) {
    const buffer = Buffer.from(encryptedData, 'base64');
    return crypto.privateDecrypt({ key: privateKey, passphrase: '' }, buffer).toString('utf-8');
}

const plainText = 'Wussup';
const encryptedData = encrypt(plainText, publicKey);
console.log('Encrypted:', encryptedData);

const decryptedData = decrypt(encryptedData, privateKey);
console.log('Decrypted:', decryptedData);
```

Digital Signature

A *digital signature* is a cryptographic technique used to verify the authenticity and integrity of digital messages or documents. It provides assurance that the content has not been altered and was created by a specific entity.



```
const crypto = require('crypto');

const privateKey= ''

function sign(data, privateKey) {
    const sign = crypto.createSign('SHA256');
    sign.write(data);
    sign.end();
    return sign.sign(privateKey, 'base64');
}

function verify(data, signature, publicKey) {
    const verify = crypto.createVerify('SHA256');
    verify.write(data);
    verify.end();
    return verify.verify(publicKey, signature, 'base64');
}

const dataToSign = 'Hello, World!';
const signature = sign(dataToSign, privateKey);
console.log('Signature:', signature);

const isSignatureValid = verify(dataToSign, signature, publicKey);
console.log('Is Signature Valid:', isSignatureValid);
```

Implementation Considerations

When implementing **asymmetric encryption** and *digital signatures*, factors such as key management, algorithm selection, and performance impact must be carefully considered to ensure robust security.

Conclusion

The implementation of **asymmetric encryption** and *digital signatures* is crucial for securing sensitive data in modern digital environments. By understanding and applying these techniques, organizations can enhance data security and protect against unauthorized access and tampering.

Thanks!