Name: Devanshu Surana
Roll No.: 23
Prn:1032210755
Panel: C batch:C1

Lab A5: Implementation of Digital Signature using DSA

Objective of Lab

1. To understand and implement Digital signature using DSA
2. To understand and ensure Digital signature provides authenticity, integrity, and non-repudiation of electronic documents and messages.

Theory

What Are Digital Signatures?

The objective of digital signatures is to authenticate and verify documents and data. This is necessary to avoid tampering and digital modification or forgery during the transmission of official documents.

With one exception, they work on the public key cryptography architecture. Typically, an asymmetric key system encrypts using a public key and decrypts with a private key. For digital signatures, however, the reverse is true. The signature is encrypted using the private key and decrypted with the public key. Because the keys are linked, decoding it with the public key verifies that the proper private key was used to sign the document, thereby verifying the signature's provenance.

A Digital Signature Algorithm (DSA) is a widely used asymmetric key algorithm for secure digital signatures. It was proposed by the National Institute of Standards and Technology (NIST) in 1991. DSA is based on the mathematical properties of modular exponentiation and the discrete logarithm problem.

DSA Algorithm provides three benefits, which are as follows:

Message Authentication: You can verify the origin of the sender using the right key combination.

Integrity Verification: You cannot tamper with the message since it will prevent the bundle from being decrypted altogether.

Non-repudiation: The sender cannot claim they never sent the message if verifies the signature.

Digital Signature Algorithm (DSA) in Cryptography: How It Works & More
By Baivab Kumar Jena
Last updated on Feb 14, 2023113522
Everything You Need to Know About DSA Algorithm
Table of Contents
What Is Asymmetric Encryption?Why is Cryptography a Smart Solution?What Are Digital Signatures?Block Diagram of Digital SignatureImportance of Digital Signature View More
The Covid-19 pandemic has given a new life to the work-from-home initiative, taking the corporate world into an untapped phase. Without a doubt, most of the users reading this have had to digitally sign some official documents over the past couple of years because of the lack of face-to-face interaction and standard distance constraints. To maintain the authenticity and integrity of such documents holding critical information, the DSA Algorithm was proposed and passed as a global standard for verifying digital signatures.

Before moving forward with the algorithm, you will get a refresher on asymmetric encryption, since it verifies digital signatures according to asymmetric cryptography architecture, also known as public-key cryptography architecture.

What Is Asymmetric Encryption?
You utilize two distinct keys in asymmetric encryption methods, one for encryption and the other for decryption. You use the public key for encryption; meanwhile, you use the private key for decryption. However, you must generate both keys from the receiver's end.

process.

Using separate keys for encryption and decryption, as seen in the figure above, has helped eliminate key exchange, as seen in the case of symmetric encryption.

For example, if Alice needs to send a message to Bob, both the private and public keys must belong to Bob.

alice.

The process for the above image is as follows:

Step 1: Alice first uses Bob's public key to encrypt the message
Step 2: The encrypted message reaches Bob
Step 3: Bob decrypts the message with his secret key
This eliminates the requirement for the sender and recipient to exchange any secret keys, minimizing the window of opportunity for exploitation.

Now that you learned how asymmetric encryption happens, you will look at how the digital signature architecture is set up.

CEH (v12) - Certified Ethical Hacking Course
Get trained on advanced methodologies hackers useVIEW COURSECEH (v12) - Certified Ethical Hacking Course
Why is Cryptography a Smart Solution?
Cryptography involves using codes and ciphers to protect sensitive information from unauthorized access. Cryptography protects data in many applications, including banking, military communications, and secure emails.

Cryptography is a smart solution because it provides an efficient and secure way to protect sensitive data and communications. Cryptography uses advanced mathematical algorithms to encrypt data so only authorized parties can access it, and it also prevents third parties from intercepting communications or manipulating data.

It enables secure transmission over the internet, protecting data from unauthorized access and potential tampering. Cryptography also allows for data authentication, which can be used to verify the integrity of data and the sender's identity, meaning that data cannot be tampered with or changed without authorization. Finally, cryptography can authenticate users and ensure that only the intended recipient can access the data.

What Are Digital Signatures?
The objective of digital signatures is to authenticate and verify documents and data. This is necessary to avoid tampering and digital modification or forgery during the transmission of official documents.

With one exception, they work on the public key cryptography architecture. Typically, an asymmetric key system encrypts using a public key and decrypts with a private key. For digital signatures, however, the reverse is true. The signature is encrypted using the private key and decrypted with the public key. Because the keys are linked, decoding it with the public key verifies that the proper private key was used to sign the document, thereby verifying the signature's provenance.

Read more: What is Cryptography And How Does It Protect Data?

ds_process-DSA_Algorithm.

M - Plaintext

H - Hash function

h - Hash digest

'+' - Bundle both plaintext and digest

E - Encryption

D - Decryption

The image above shows the entire process, from the signing of the key to its verification. So, go through each step to understand the procedure thoroughly.

Step 1: M, the original message is first passed to a hash function denoted by H# to create a digest.
Step 2: Next, it bundles the message together with the hash digest h and encrypts it using the sender's private key.
Step 3: It sends the encrypted bundle to the receiver, who can decrypt it using the sender's public key.
Step 4: Once it decrypts the message, it is passed through the same hash function (H#), to generate a similar digest.
Step 5: It compares the newly generated hash with the bundled hash value received along with the message. If they match, it verifies data integrity.
There are two industry-standard ways to implement the above methodology. They are:

RSA Algorithm
DSA Algorithm
Both the algorithms serve the same purpose, but the encryption and decryption functions differ quite a bit. So, now that you understand how it is supposed to function while verifying the signature, let's deep dive into our focus for today, the DSA Algorithm.

Block Diagram of Digital Signature
A digital signature is a type of electronic signature based on cryptography and used to authenticate the identity of the sender of a message or the signer of a document and to ensure that the original content of the message or document is unchanged.

DS_Block_Diag

The block diagram of the digital signature process is as follows:

Message or Document: This is the data or document that needs to be signed.
Hashing: The document or message is converted into a fixed-length hash using a cryptographic hash function.

Encryption: The hash is then encrypted using the sender's private key.
Signature: The encrypted hash is the document's digital signature.
Verification: The signature is then verified using the sender's public key to ensure the signature's validity and the document's authenticity.
CEH (v12) - Certified Ethical Hacking Course
Get trained on advanced methodologies hackers useVIEW COURSECEH (v12) - Certified Ethical Hacking Course
Explanation of the Block Diagram
The digital signature block diagram outlines the process of creating and verifying digital signatures.

The process begins with a message being sent from one party to another. This message is then run through a cryptographic hashing algorithm, which produces a hash of the message. Then, the hash is encrypted using the sender's private key. The encrypted hash is then attached to the original message, forming the digital signature.
After that, the recipient verifies the digital signature by decrypting the hash with the sender's public key and then comparing it to a new hash generated from the original message. This decrypting process assures that the message has not been altered in transit and can be trusted as coming from the sender. Once again, the digital signature proves that the data is from a trusted source and has not been changed.
Importance of Digital Signature
The importance of digital signatures is not negligible in this digital world, and there are some,

Ensures Authenticity: Digital signatures ensure the authenticity of a message or transaction by proving that the message or signature was created using the private key associated with the digital signature.
Offers Non-repudiation: A digital signature provides an entire record that a specific individual signed the document or transaction at a particular time. This feature prevents the individual from denying that they signed it.
Provides Security: Digital signatures use encryption algorithms to protect the data from unauthorized access and tampering. The cryptographic techniques used by digital signatures also protect the data from being changed or manipulated during transmission.
Improves Efficiency: Digital signatures can reduce the time and money spent on paperwork, printing, scanning, and mailing documents.
Enhances Compliance: Digital signatures help organizations to meet regulatory and legal compliance requirements by providing an audit trail of signed documents.
Role of Digital Signatures
Digital signatures are used for authentication or verification. They are used to verify the authenticity and integrity of a digital document or message.
The prominent role of a digital signature is to ensure the integrity and authenticity of a digital document or message. A digital signature is essentially a type of electronic signature which provides non-repudiation and authentication between two parties. It serves as proof that the

two parties have agreed to specific terms and conditions and have mutually verified the contents of a document or message.
Digital signatures can also protect confidential information, ensuring that only authorized persons can access the data.
Furthermore, digital signatures can also be used to track a digital transaction's source and ensure that it has not been modified.
Encryption With Digital Signature
Encryption is transforming information (referred to as plaintext) using an algorithm (called cipher) to make it unreadable to anyone except those possessing special knowledge, usually referred to as a key. The process's result is encrypted information. In cryptography, this encrypted information is referred to as ciphertext.

What Is the DSA Algorithm?
Digital Signatures Algorithm is a FIPS (Federal Information Processing Standard) for digital signatures. It was proposed in 1991 and globally standardized in 1994 by the National Institute of Standards and Technology (NIST). It functions on the framework of modular exponentiation and discrete logarithmic problems, which are difficult to compute as a force-brute system.

DSA Algorithm provides three benefits, which are as follows:

Message Authentication: You can verify the origin of the sender using the right key combination.
Integrity Verification: You cannot tamper with the message since it will prevent the bundle from being decrypted altogether.
Non-repudiation: The sender cannot claim they never sent the message if verifies the signature.
dsa-DSA_Algorithm

The image above shows the entire procedure of the DSA algorithm. You will use two different functions here, a signing function and a verification function. The difference between the image of a typical digital signature verification process and the one above is the encryption and decryption part. They have distinct parameters, which you will look into in the next section of this lesson on the DSA Algorithm.

CEH (v12) - Certified Ethical Hacking Course
Get trained on advanced methodologies hackers useVIEW COURSECEH (v12) - Certified Ethical Hacking Course
Steps in DSA Algorithm
Keeping the image above in mind, go ahead and see how the entire process works, starting from creating the key pair to verifying the signature at the end.

1. Key Generation
There are two steps in the key generation process: parameter generation and per-user keys.

Parameter Generation

- Initially a user needs to choose a cryptographic hash function (H) along with output length in bits |H|. Modulus length N is used in when output length |H| is greater.
- Then choose a key length L where it should be multiple of 64 and lie in between 512 and 1024 as per Original DSS length. However, lengths 2048 or 3072 are recommended by NIST for lifetime key security.
- The values of L and N need to be chosen in between (1024, 60), (2048, 224), (2048, 256), or (3072, 256) according to FIPS 186-4. Also, a user should chose modulus length N in such a way that modulus length N should be less than key length (N<L) and less than and equal to output length (N<=|H|).
- Later a user can choose a prime number q of N bit and another prime number as p of L bit in such a way that p-1 is multiple of q.
- And then choose h as an integer from the list (2……..p-2).

Once you get p and q values, find out
g = h^(p-1)/q*mod(p). If you get g = 1, please try another value for h and compute again for g except 1.

p, q and g are the algorithm parameters that are shared amongst different users of the systems.

Per-user Keys
To compute the key parameters for a single user, first choose an integer x (private key) from the list (1…….q-1), then compute the public key, y=g^(x)*mod(p).

2. Signature Generation

- It passes the original message (M) through the hash function (H#) to get our hash digest(h).
- It passes the digest as input to a signing function, whose purpose is to give two variables as output, s, and r.
- Apart from the digest, you also use a random integer k such that 0 < k < q.
- To calculate the value of r, you use the formula r = (gk mod p) mod q.
- To calculate the value of s, you use the formula s = [K-1(h+x . R)mod q].
- It then packages the signature as {r,s}.
- The entire bundle of the message and signature {M,r,s} are sent to the receiver.

3. Key Distribution
While distributing keys, a signer should keep the private key (x) secret and publish the public key (y) and send the public key (y) to the receiver without any secret mechanism.
Signing of message m should be done as follows:

- first choose an integer k from (1……q-1)
- compute
  r = g^(k)*mod(p)*mod(q). If you get r = 0, please try another random value of k and compute again for r except 0.

- Calculate
  s=(k^(-1)*(H(m)+xr))*mod(q). If you get s = 0, please try another random value of k and compute again for s except 0.

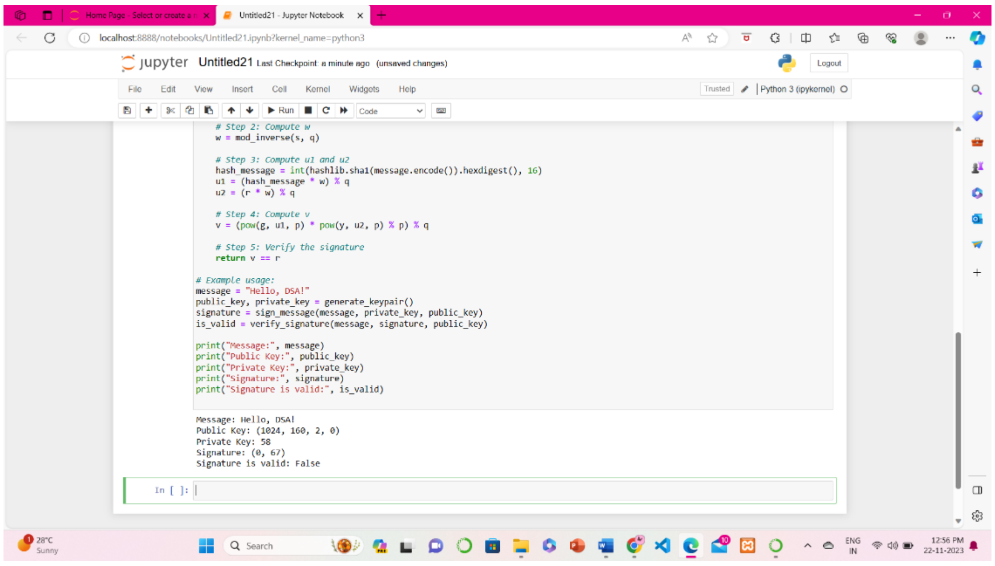
The signature is defined by two key elements (r,s). Also, key elements k and r are used to create a new message. Nevertheless, computing r with modular exponential process is a very expensive process and computed before the message is known. Computation is done with the help of the Euclidean algorithm and Fermat's little theorem.
4. Signature Verification

- You use the same hash function (H#) to generate the digest h.
- You then pass this digest off to the verification function, which needs other variables as parameters too.
- Compute the value of w such that: s*w mod q = 1
- Calculate the value of u1 from the formula, u1 = h*w mod q
- Calculate the value of u2 from the formula, u2 = r*w mod q
- The final verification component v is calculated as v = [((gu1 . yu2) mod p) mod q].
- It compares the value of v to the value of r received in the bundle.
- If it matches, the signature verification is complete.
- Having understood the functionality of the DSA Algorithm, you must know the advantages this algorithm offers over alternative standards like the RSA algorithm.

Program

```
import hashlib
import random
from sympy import mod_inverse

def generate_keypair():
    q = 160
    p = 1024
    g = 2
    x = random.randint(2, q - 1)


    y = pow(g, x, p)

    public_key = (p, q, g, y)
    private_key = x

    return public_key, private_key

def sign_message(message, private_key, public_key):
    p, q, g, y = public_key

    k = random.randint(2, q - 1)

    r = pow(g, k, p) % q

    # Step 4: Compute s
    hash_message = int(hashlib.sha1(message.encode()).hexdigest(), 16)
    s = (mod_inverse(k, q) * (hash_message + private_key * r)) % q

    return (r, s)

def verify_signature(message, signature, public_key):
    p, q, g, y = public_key
    r, s = signature


    if not (0 < r < q and 0 < s < q):
        return False


    w = mod_inverse(s, q)

    hash_message = int(hashlib.sha1(message.encode()).hexdigest(), 16)
    u1 = (hash_message * w) % q
    u2 = (r * w) % q

    v = (pow(g, u1, p) * pow(y, u2, p) % p) % q
    return v == r
message = "Hello, DSA!"
public_key, private_key = generate_keypair()
signature = sign_message(message, private_key, public_key)
is_valid = verify_signature(message, signature, public_key)

print("Message:", message)
print("Public Key:", public_key)
print("Private Key:", private_key)
print("Signature:", signature)
print("Signature is valid:", is_valid)
```

Output Screen shots



```
    # Step 2: Compute w
    w = mod_inverse(s, q)

    # Step 3: Compute u1 and u2
    hash_message = int(hashlib.sha1(message.encode()).hexdigest(), 16)
    u1 = (hash_message * w) % q
    u2 = (r * w) % q

    # Step 4: Compute v
    v = (pow(g, u1, p) * pow(y, u2, p) % p) % q

    # Step 5: Verify the signature
    return v == r

# Example usage:
message = "Hello, DSA!"
public_key, private_key = generate_keypair()
signature = sign_message(message, private_key, public_key)
is_valid = verify_signature(message, signature, public_key)

print("Message:", message)
print("Public Key:", public_key)
print("Private Key:", private_key)
print("Signature:", signature)
print("Signature is valid:", is_valid)
```

```
Message: Hello, DSA!
Public Key: (1024, 160, 2, 0)
Private Key: 58
Signature: (0, 67)
Signature is valid: False
```

Conclusion: Thus, we have learned and implemented the DSA algorithm.

FAQs:

1. What is Digital Signature? What are pros and cons of digital signature?
2. What is a Digital Signature Certificate (DSC)?
3. Are Digital Signature Certificate (DSC)s legally valid in India?

References: