**Experiment No: 8**

**Aim:** Write a program to sign and verify a document using DSA algorithm

**Theory/ Discussion**:

 The digital signature is a mechanism that verifies the authority of digital messages as well as documents. It is very popular because it provides more security than other signatures. In Java, JDK Security API is used to create and implement digital signatures. In this section, we will discuss the digital signature mechanism and also implement the digital signature mechanism in a Java program.

The **digital signature** is an electronic signature to sign a document, mail, messages, etc. It validates the **authenticity,** and **integrity** of a message or document. It is the same as a handwritten signature, seal, or stamp. It is widely used to verify a digital message, financial documents, identity cards, etc.

In short, we can say that it ensures the following:

* **Integrity:** It ensures the message or a document cannot be altered while transmitting.
* **Authenticity:** The author of the message is really who they claim to be.
* **Non-repudiation:** The author of the message can't later deny that they were the source.

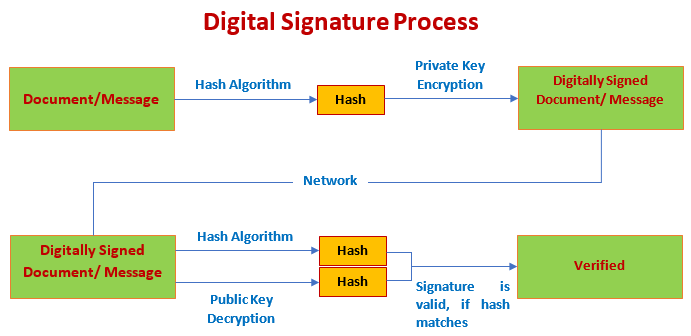
What Is the DSA Algorithm?

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.



Digital Signature Algorithm (DSA) in Cryptography: How It Works and Advantages

By Simplilearn

Last updated on Aug 23, 202242321

Everything You Need to Know About DSA Algorithm

Table of Contents

What Is Asymmetric Encryption?What Are Digital Signatures?What Is the DSA Algorithm?Steps in DSA AlgorithmCorrectness of the AlgorithmView 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.

PGP in Cyber Security With Modules From MIT SCC

Your Cyber Security Career Success Starts Here!VIEW COURSEPGP in Cyber Security With Modules From MIT SCC

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.

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.

Cybersecurity Bootcamp

Certificate and Masterclasses From UCI DCEEXPLORE NOWCybersecurity Bootcamp

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.

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.

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

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.

**Examples:**

**Steps/Method/Code:**

1. **import** java.io.\*; //input the file data to be signed
2. **import** java.security.\*; //provides methods for signing the data
3. **public** **class** GenerateDigitalSignature
4. {
5. **public** **static** **void** main(String args[])
6. {
7. /\* Generate a DSA signature \*/
8. **if** (args.length != 1)
9. {
10. System.out.println("Usage: nameOfFileToSign");
11. }
12. **else** **try**
13. {
14. // the rest of the code goes here
15. }
16. **catch** (Exception e)
17. {
18. System.err.println("Caught exception " + e.toString());
19. }
20. }
21. }

**VerifyDigitalSignature.java**

1. **import** java.io.\*;
2. **import** java.security.\*;
3. **import** java.security.spec.\*;
4. **public** **class** VerifyDigitalSignature
5. {
6. **public** **static** **void** main(String args[])
7. {
8. /\* Verify a DSA signature \*/
9. **if** (args.length != 3) {
10. System.out.println("Usage: VerifyDigitalSignature " +"publickeyfile signaturefile " + "datafile");
11. }
12. **else** **try**
13. {
14. // the rest of the code goes here
15. }
16. **catch** (Exception e)
17. {
18. System.err.println("Caught exception " + e.toString());
19. }
20. }
21. }

**Learning Outcomes:**

With this, you have understood the importance of asymmetric cryptography, the working of digital signatures, the functionality of DSA, the steps involved in the signature verification, and its advantages over similar counterparts.

**Reference**

<https://www.javatpoint.com/java-digital-signature>

<https://www.simplilearn.com/tutorials/cryptography-tutorial/digital-signature-algorithm>