The Digital Signature Standard

Summary

Introduction

This research paper was proposed by NIST (National Institute of Standards and Technology) and it aims at building an algorithm which can be used to make digital signatures, instead of traditional written signatures. The Digital Signature can be used to verify both the identity and authenticity of the originator. The name of the Algorithm is Digital Signature Algorithm (DSA). It is a pair of public and private keys. DSA comprises of two parts

- **Signature Generation**: by using **private** key.
- **Signature Verification**: by using **public** key.

Implementation

Every user has their own set of private & public key. Private keys are meant to keep confidential. Whereas, public keys are open to all users. Signature Generation involves using a **hash function H**, which helps in obtaining a condensed version of data, often known as **message digest**. This digest is then signed by the sender/originator by using their private key. (Who does not know the private key, cannot sign the document. Thus, signatures cannot be forged). The digital signature is then sent to the receiver, along with the signed data. The receiver then validates the signature by the public key of the sender. Both Signature Generation and Verification use hash function H.

That is how DSA authenticates the integrity of the data and identifies the sender. Not only this, the DSA can also be used to prove third-party that this data was signed by its signing authority.

This Algorithm can be used at the e-platforms where data interchanges between authorities, and where the authentication of its generating source and assurance of its integrity, is at utmost priority. For e.g. E-mails, electronic data interchange, etc.

To bind the user's identity and its public keys, a mutually trusted party known as **Certifying Authority** (CA) is responsible to form a certificate.

DSA Parameters:

p = a prime modulus where $2^{511} < 2^{512}$

q = a prime divisor of p - 1 , where $2^{159} < q < 2^{160}$.

g = $h^{(p-1)/q}$ mod p, where h is any integer with $0 < h < p$ such that $h^{(p-1)/q}$ mod p > 1.

Private Key → **x** = an integer with $0 < x < q$.

Public Key → **y** = g^x mod p.

m = the message to be signed and transmitted.

k = a random integer with $0 < k < q$.

H = a one-way hash function.

The integers **p**, **q** and **g** can be public and can be common to a group of users. **x** and **k** must be secret. **k** must be changed for each signature.

Signature Generation & Verification:

To generate and sign the message **m** user chooses a random integer k and computes:

r = (g^k mod p) mod q

s = (k^{-1}(H(m) + xr)) mod q

The values **r** and **s** constitute the signature of the message. These are transmitted along with the message **m** to the recipient.

m', **r'**, and **s'** be the **received versions** of **m**, **r**, and **s**, respectively, and let **y** be the **public key** of the sender.

To verify the signature, the recipient first checks to see that **0 < r' < q** and **0 < s' < q**; **if either condition is violated the signature is rejected**. If these two conditions are satisfied, the recipient computes

w = $(s')^{-1}$ mod q

u1 = ((H(m'))w) mod q

u2 = ((r')w) mod q

v = $(((g)^{u1}(y)^{u2})$ mod p) mod q.

If v = r', the signature is verified

If v does not equal r', the signature is not valid.

Proof v = r'

The purpose is to prove that we get v = r', when m'=m, r'=r, s'=s.

This is proved by **proving 4 Lemmas :**

Lemma 1 $g^t \bmod p = g^{t \bmod q} \bmod p$	**Lemma 2** $g^{a \bmod q + b \bmod q} \bmod p = g^{(a+b) \bmod q} \bmod p$
Lemma 3 $y^{(rw) \bmod q} \bmod p = g^{(xrw) \bmod q} \bmod p$	**Lemma 4** $((H(m) + xr)w) \bmod q = k$

Lemma 1	**Lemma 2**
Proof: By the Euler/Fermat theorem, since h is relatively prime to p, we have $h^{p-1} \bmod p = 1$. Hence for any nonnegative integer n, $g^{nq} \bmod p = (h^{(p-1)/q} \bmod p)^{nq} \bmod p$ $= h^{((p-1)/q)nq} \bmod p$ $= ((h^{(p-1)} \bmod p)^n)$ $= 1^n \bmod p$ $= 1.$ Thus, for any nonnegative integers **n** and **z,** $g^{nq+z} \bmod p = (g^{nq}\, g^z) \bmod p$ $= ((g^{nq} \bmod p)\,(g^z \bmod p)) \bmod p$ $= g^z \bmod p.$ Any nonnegative integer **t** can be represented uniquely as **t = nq + z** where **n** and **z** are nonnegative integers. Then **$g^t \bmod p = g^{t \bmod q} \bmod p$**. Also, z = t mod q. Hence Proved.	Proof: By Lemma 1 we have $g^{a \bmod q + b \bmod q} \bmod p = g^{(a \bmod q + b \bmod q) \bmod q} \bmod p$ $= g^{(a+b) \bmod q} \bmod p$ **$g^{a \bmod q + b \bmod q} \bmod p = g^{(a+b) \bmod q} \bmod p$** Hence Proved.

Lemma 3

Proof: Since $y = g^x \bmod p$, using Lemma 1

$$\begin{aligned} y^{(rw) \bmod q} \bmod p &= (g^x \bmod p)^{(rw) \bmod q} \bmod p \\ &= g^{x((rw) \bmod q)} \bmod p \\ &= g^{(x((rw) \bmod q) \bmod q} \bmod p \\ &\quad \text{(by Lemma 1)} \\ &= g^{(xrw) \bmod q} \bmod p. \end{aligned}$$

$$\mathbf{y^{(rw) \bmod q} \bmod p = g^{(xrw) \bmod q} \bmod p}$$

Hence Proved.

Lemma 4

Proof: We have

$$s = (k^{-1}(H(m) + xr)) \bmod q$$

Since $(k\,k^{-1}) \bmod q = 1$,

$$\begin{aligned} (ks) \bmod q &= (k((k^{-1}(H(m) + xr)) \bmod q)) \bmod q \\ &= ((k(k^{-1}\,(H(m) + xr)))) \bmod q \\ &= (((k\,k^{-1}) \bmod q)\,((H(m) + xr) \bmod q)) \bmod q \\ &= (H(m) + xr) \bmod q \end{aligned}$$

Since $\mathbf{w = s^{-1} \bmod q}$ we have $(ws) \bmod q = 1$, and thus

$$\begin{aligned} ((H(m) + xr)w) \bmod q &= (((H(m) + xr) \bmod q)\,(w \bmod q)) \bmod q \\ &= (((ks) \bmod q)\,(w \bmod q)) \bmod q \\ &= (kws) \bmod q = ((k \bmod q)\,((ws) \bmod q)) \bmod q \\ &= k \bmod q \end{aligned}$$

Since $0 < k < q$ we have $\mathbf{k \bmod q = k}$.

$$\mathbf{((H(m) + xr)w) \bmod q = k}$$

Hence Proved.

Random Number Generation for the DSA

- Any implementation of the DSA requires the ability to generate random numbers.
- Random numbers are used to derive a **user's private key 'x'** and a **user's per-message-random-secret-number**, **'k'**.
- These random numbers are selected to be between **0** and the **160**-bit prime q (as specified in the standard).
-

Modular Arithmetic for the DSA

To use the DSS, one way suggested by this research paper is the development of a modular arithmetic package suited to the processor, one intend to use.

They have talked about **2 type of processors :**

- **Hardware**
- **Software**

For Hardware implementations, the reference given is of "**VLSI Implementation of Public Key Encryption Algorithm**s" by Orton, Roy, Scott, Peppard and Tavares from the Proceedings of CRYPTO=86.

For Software implementations, the reference given is of **“A Cryptographic Library for the Motorola DSP56000"** by Dusse and Kaliski from the Proceedings of EUROCRYPT-90 and **"Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor"** by Barrett from the Proceedings of CRYPTO-86.