A MODIFIED DIGITAL SIGNATURE STANDARD (DSS) ALGORITHM TO IMPROVE NETWORK SECURITY

Omer Ali Beshir*, Elyas Towhid Sheik Mahmoud*, Shareeful Islam*

Abstract

Authentication is fundamentally a part of our lives as privacy. We use authentication throughout everyday lives; when we sign our name to some document or we move to a world where our decisions and agreements are communicate electronically, we need to have electronic techniques for providing authentication. Digital signatures are used to detect unauthorized modification to data and to authenticate the identity of the signatory. In this research paper, a digital signature algorithm is developed which is a modified version of the Digital Signature Algorithm (DSA) proposed by the National Institute of Standards and Technology (NIST).

Introduction

A digital signature is represented in a computer as a string of binary digits. A digital signature is computed using a set of rules and a set of parameters such that the identity of the signatory and integrity of the data can be verified. An algorithm provides the capability to generate and verify signatures. Signature generation [1] makes use of a private key to generate a digital signature. Signature verification makes use of a public key, which corresponds to, but is not the same as, the private key. Each user possesses a private and public key pair. Public keys are assumed to be known to the public in general. Private keys are never shared. Anyone can verify the signature of a user by employing that user's public key. Signature generation can be performed only by the possessor of the user's private key. A hash function is used in the signature generation process to obtain a condensed version of data, called a message digest (see Figure 1). The message digest is then input to the digital signature algorithm (DSA) to generate the digital signature. The digital signature is sent to the intended verifier along with the signed data (often called the message). The verifier of the message and signature verifies the signature by using the sender's public key. The same hash function must also be used in the verification process. The hash function is specified in a separate standard.

Applications of Digital Signature

Because the DSA authenticates both the identity of the signer and the integrity of the signed information, it can be used in a variety of applications. For example, the DSA could be utilized in an electronic mail system [8]. After a party generated a message, that party could sign it using the party's private key. The signed message could then be sent to a second party. After verifying the received message, the second party would have confidence that the first party

*Department of Computer Science and Information Technology, Islamic University of Technology
Board Bazar, Gazipur -1704, Bangladesh.

signed the message. The second party would also know that the message was not altered after the first party signed it. An electronic time stamp could be affixed to documents in electronic form and then signed using the DSA. Applying the DSA to the document would protect and verify the integrity.

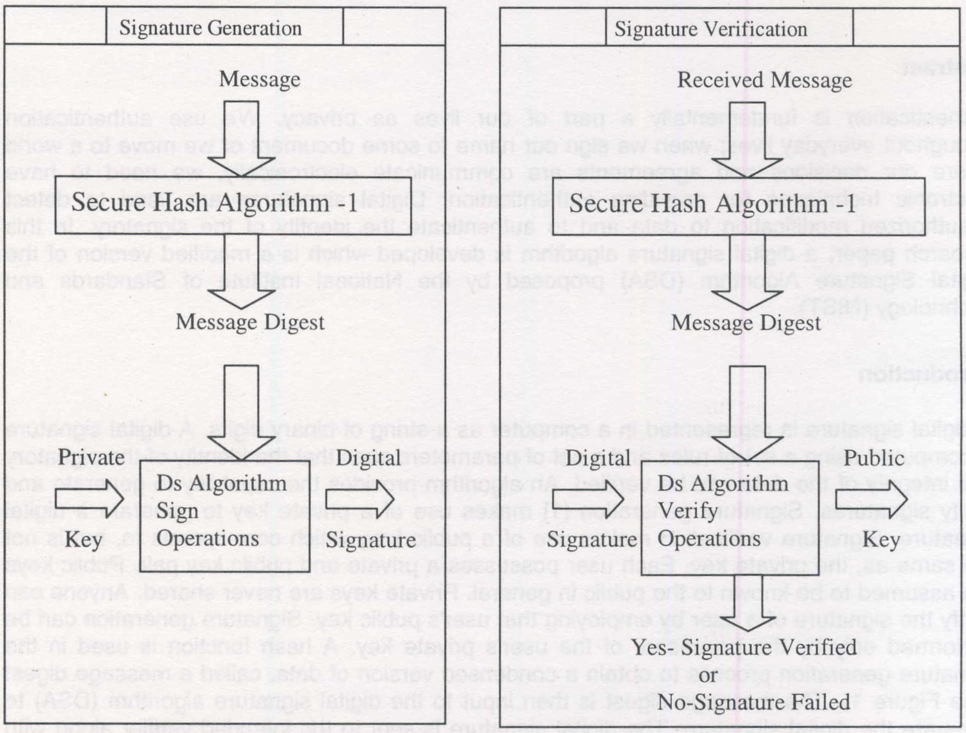

Figure 1 : A Conventional Signal Generation and Verification process in DSS

The DSA could be useful in the distribution of software. A digital signature could be applied to software after it has been validated and approved for distribution. Before installing the software on a computer, the signature could be verified to be sure no unauthorized changes (such as the addition of a virus) have been made. The digital signature could be verified periodically to ensure the integrity of the software.

In database applications [7], the integrity of information stored in the database is often essential. The DSA could be employed in a variety of database applications to provide integrity. For example, information could be signed when it was entered into the database. To maintain integrity, the system could also require that all updates or modifications to the information be signed. Before signed information was viewed by a user, the signature could be verified. If the signature verified correctly, the user would know the information had not been altered by an unauthorized party. The system could also include signatures in the audit information to provide a record of users who modified the information.

Digital Signature Algorithm (DSA)

A digital signature algorithm [6] is used by a *signatory* to generate a digital signature on data and by a *verifier* to verify the authenticity of the signature. Each *signatory* has a public and private key. The private key is used in the signature generation process and the public key is used in the signature verification process. For signature generation and verification, the data, which is referred to as a message, M, is reduced by means of the Secure Hash Algorithm (SHA-1) specified in FIPS 180-1. An adversary, who does not know the private key of the signatory, cannot generate the correct signature of the signatory. In other words, signatures cannot be forged. However, by using the *signatory's* public key, anyone can verify a correctly signed message. A means of associating public and private key pairs to the corresponding users is required. That is, there must be a binding of a user's identity and the user's public key. A mutually trusted party may certify this binding. For example, a certifying authority could sign credentials containing a user's public key and identity to form a certificate.

Hash function

A hash function H [4] is a transformation that takes an input m and returns a fixed-size string, which is called the hash value h (that is, h = H (m)). Hash functions with just this property have a variety of general computational uses, but when employed in cryptography, the hash functions are usually chosen to have some additional properties.
The basic requirements for a cryptographic hash function are as follows.

- The input can be of any length.
- The output has a fixed length.
- H (x) is relatively easy to compute for any given x.
- H (x) is one-way.
- H (x) is collision-free.

A hash function H is said to be *one-way* if it is hard to invert, where ``hard to invert'' means that given a hash value h, it is computationally infeasible to find some input x such that H (x) = h. If, given a message x, it is computationally infeasible to find a message y not equal to x such that H(x) = H(y), then H is said to be a *weakly collision-free* hash function. A *strongly collision-free* hash function H is one for which it is computationally infeasible to find any two messages x and y such that H (x) = H (y).

Damgard and Merkle [3] greatly influenced cryptographic hash function design by defining a hash function in terms of what is called a *compression function*. A compression function takes a fixed-length input and returns a shorter, fixed-length output. Given a compression function, a hash function can be defined by repeated applications of the compression function until the entire message has been processed. In this process, a message of arbitrary length is broken into blocks whose length depends on the compression function, and " padded " (for security reasons) so the size of the message is a multiple of the block size. These blocks are then processed sequentially, taking as input the result of the hash so far and the current message block, with the final output being the value the message (Figure : 2).

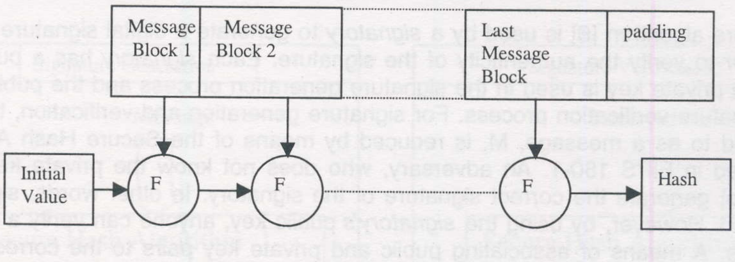

Figure : 2 Damgard and Merkle iterative structure for hash functions; F is a compression function.

Review of the DSA proposed by National Institute of Standards and Technology

A digital signature algorithm scheme is a method by which the signer can sign an electronic documents for the receiver (or the verifier) to keep as an evidence that the document was indeed sent originally from the signer. The National Institute of Standards and Technology (NIST) has proposed the Digital Signature Algorithm (DSA)[3] as the public standard for digital signature. Brief review of the DSA algorithm is given below. In the DSA, a trusted center (TC) is assumed and the role of the TC is to select and publish system parameters for public usage.

System parameters:

p: a prime modulus where $2^{511} < p < 2^{512}$
q: a prime divisor of (p-1) where $2^{159} < q < 2^{160}$
g: $g = h^{(p-1)/q}$ (mod p) > 1 where h is a random integer with 1< h < p (i.e. g has order q (mod p)).

Public key and secret key of users:

x : each users selects an integer x as his secret key with 1 < x < q
y : corresponding to x, user computes his public key y such that y = gx(mod p) where 1<y < p

Signature generation and Verification for message m:

Generation:

The signer computes
r = (g^k(mod p))(mod q) where k is a random integer such that 1 < k < q.
s = (k^{-1} (H(m) + x X r))(mod q) where H( ) is one way hash function.
The pair of numbers (r, s) constitutes the signature of the message m, signed by the user with public key y and secret key x.

Verification:

To verify the signature pair (r, s), the recipient computes $w = s^{-1}$ (mod q),

$u_1 = w$ X H(m)(mod q)

$u_2 = w$ X r (mod q), then he verifies the validity of the following equation

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

A modified Digital Signature Algorithm

A new modified digital Signature algorithm based on the DSA is developed in which the signature signer can perform more efficiently with the elimination of one modular inverse computation, i.e. k^{-1} (mod q). There are two approaches [9] for the above modular inverse computation. One based on Fermat's Theorem, the other based on the extended Euclidean algorithm. The former is a modular exponentiation, therefore the computation is more simple while less efficient. Theoretically using the square multiply algorithm, the former algorithm takes approximately 1.5 X $\log_2 q$ modular multiplication where q is the modulus. The latter is faster but the algorithm requires more intermediate variables and is somewhat more complex. Knuth shows that the average number of iterations performed by the algorithm is approximately 0.843 X ln (q) + 1.47. In each iteration, one division and three multiplications are required.

Signature generation & verification for message m in the modified algorithm

Generation:

$r = (g^k$ (mod p))(mod q)

s = (r X k - H(m)) X x^{-1}(mod q)

The pair of numbers (r, s) constitute the signature of the message m signed by the user with public key y and secret key x. For any user, the secret key x is fixed, then its modular inverse x^{-1}(mod q) can be computed in advance and used for each signature generation.

Verification:

To verify the signature pair (r , s), the recipient computes

$w = r^{-1}$(mod q),

$u_1 = w$ X H(m) (mod q)

$u_2 =$ w X s (mod q)

Then he verifies the validity of the following equation

r = (g^{u1} X y^{u2}(mod p)) (mod q)

Using the modified algorithm, a small device can be designed such that the modular inverse computation can both be eliminated during signing and verifying the DSA signatures.

Conclusion

Many Organizations are transforming paper-based systems into automated electronic systems in order to reduce costs and increase productivity. This trend has brought about a need for a reliable, cost-effective way to replace a handwritten signature with a digital signature. A digital signature can also be used to verify that information has not been altered after it is signed. So security of digital signature algorithm is a crucial part that has to be maintained. In all exists D.S algorithms signature generation and verification process is time consume and complex. On the contrary by eliminating some mathematical equations from the exists algorithm modified algorithm proposed. Complexity and time consumption can be reduced in the modified algorithm.

References

[1] William Stallings.: Data and Computer Communications, Sixth Edition, Prince Hall, Upper Saddle River, New Jersey 07458.

[2] W. Richard Stevens.: Unix Network programming, International Edition, Price Hall Software Series.

[3] K.S McCurley, "An open comment letter from the Sandia National Laboratories on the DSA of the NIST," Nov7, 1991].

[4] D.E. Denning, Cryptography and Data Security, Reading, Mass: Addison-Wesley, 1982.

[5] D.E Knuth, The Art of Computer Programming, Vol 2: Semi numerical Algorithms. Reading, Mass.: Addison-wisely, 1996.

[6] William Stallings.: Cryptography and Network Security, Principles and Practice, Second Edition,Low Price Edition, Chapter 3,4,5,9.

[7] Cooper, J. Computer and Communications Security: Strategies for the 1990s.McGraw-Hill.

[8] Comer,D.: Internetworking with TCP/IP,Volume 1: Principles, Protocol and Architecture, Prentice Hall.

[9] Popek,G., and Kline, C.:Encryption and Secure Computer Networks. ACM Computing Surveys, December 1998.