International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 12, December - 2013

Design and Implementation of Digital Signatures

K. Ramya
Assistant Professor, Department of Mathematics,
Kingston Engineering College, Katpadi, Vellore

K. Suganya
Associate Professor, Department of Software Engineering &IT[PG],
A.V.C College of Engineering, Mannampandal, Mayiladuthurai.

Abstract:

When a message is received, the recipient may desire to verify that the message has not been altered in transit. Furthermore, the recipient may wish to be certain of the originator's identity. Both of these services can be provided by the DSA. A digital signature is an electronic analogue of a written signature in that the digital signature can be used in proving to the recipient or a third party that the message was, in fact, signed by the originator. Digital signatures may also be generated for stored data and programs so that the integrity of the data and programs may be verified at any later time.

Key Words: DSA, NIST, SHA, RSA

1. Introduction

The Digital Signature Algorithm (DSA) is a United States Federal Government standard or FIPS for digital signatures. It was proposed by the National Institute of Standards and Technology (NIST) in August 1991 for use in their Digital Signature Standard (DSS), specified in FIPS 186 adopted in 1993. A minor revision was issued in 1996 as FIPS 186-1, and the standard was expanded further in 2000 as FIPS 186-2.DSA is covered by U.S. Patent 5,231,668 , filed July 26, 1991, and attributed to David W. Kravitz, a former NSA employee.

When a message is received, the recipient may desire to verify that the message has not been altered in transit.
Furthermore, the recipient may wish to be certain of the originator's identity. Both of these services can be provided by the DSA.

A digital signature is an electronic analogue of a written signature in that the digital signature can be used in proving to the recipient or a third party that the message was, in fact, signed by the originator. Digital signatures may also be generated for stored data and programs so
that the integrity of the data and programs may be verified at any later time.

2. Digital Signatures

Digital signatures work by using somebody's secret key to sign some arbitrary data. Then anybody else could use the public key of that person to verify the signature. Since the data may be arbitrary it is not suitable input to a cryptographic digital signature algorithm. For this reason and also for performance cryptographic hash algorithms are used to preprocess the input to the signature algorithm. This works as long as it is difficult enough to generate two different messages with the same hash algorithm output. In that case the same signature could be used as a proof for both messages. Nobody wants to sign an innocent message of donating 1 € to Greenpeace and find out that he donated 1.000.000 € to Bad Inc.

For a hash algorithm to be called cryptographic the following three requirements must hold

* Preimage resistance. That means the algorithm must be one way and given the output of the hash function H(x), it is impossible to calculate x.
* 2nd preimage resistance. That means that given a pair x,y with y=H(x) it is impossible to calculate an x' such that y=H(x').
* Collision resistance. That means that it is impossible to calculate random x and x' such H(x')=H(x).

The last two requirements in the list are the most important in digital signatures. These protect against somebody who would like to generate two messages with the same hash output. When an algorithm is considered broken usually it means that the Collision resistance of the algorithm is less than brute force. Using the birthday paradox the brute force attack takes 2^ ((hash size) / 2) operations

There has been a cryptographic result for the SHA-1 hash algorithms as well, although they are not yet critical. Before 2004, MD5 had a presumed collision strength of 2^64, but it has been showed to have a collision strength well under 2^50. As of November 2005, it is believed that SHA-1's collision strength is around 2^63. We consider this sufficiently hard so that we still support SHA-1. We anticipate that SHA-256/386/512 will be used in publicly-distributed certificates in the future. When 2^63 can be considered too weak compared to the computer power available sometime in the future, SHA-1 will be disabled as well. The collision attacks on SHA-1 may also get better, given the new interest in tools for creating them.

Key Generation

Key generation has two phases. The first phase is a choice of algorithm parameters, which may be shared between different users of the system

- Choose a cryptographic hash function *H*. In the original DSS, *H* was always SHA-1, but stronger hash functions from the SHA family are also in use. Sometimes the output of a newer hash function is truncated to the size of an older one for compatibility with existing key pair.
- Decide on a key length L. This is the primary measure of the cryptographic strength of the key. The original DSS constrained L to be a multiple of 64 between 512 and 1024.
- That L Should always be 1024.Lter yet, NIST 800-57 recommends lengths of 2048(or 3072) for keys with security lifetimes extending beyond 2010(0r 2030), using correspondingly longer hashes and q.
- Choose a prime q with the same number of bits as the output of H.
- Choose a L bit prime p such p-1 is a multiple of q.
- Choose g, a number whose multiplicative order modulo p is q.This may be done by setting $g = h^{(p-1)/q} \bmod p$ for some arbitrary h(1< h< p-1), and trying again if the result comes out as 1.

The algorithm parameters (p, q, g) may be shared between different users of the system. The second phase computes private and public keys for a single user:

- Choose x by some random method, where $0 < x < q$.
- Calculate $y = g^x \bmod p$.
- Public key is (p, q, g, y).Private key is x.

The forthcoming FIPS 186-3,SHA-224/256/384/512 as the hash function, q of size 224 and 256 bits, and L equal to 2048 and 3072, respectively.

There exist efficient algorithms for computing the modular exponentiations $h^a \bmod p$ and $g^x \bmod p$.

Signing

- Generate a random per-message value k where $0 < k < q$.

- Calculate r = (g^x mod p) mod q
- Calculate s = (k^{-1} (H(m) + x * r)) mod q
- Recalculate the signature in the unlikely case that r = 0 or s = 0
- The signature is (r, s).

The extended Euclidean algorithm can be used to compute the modular inverse k^{-1} mod q.

Verifying

- Reject the signature if either 0 < r < q or 0 < s < q is not satisfied.
- Calculate w = (S) -1 mod q.
- Calculate u1 = (H(m) * w) mod q.
- Calculate u2 = (r * w) mod q.
- Calculate v = ((g^{u1} * y^{u2}) mod p) mod q
- The signature is valid if v = r.

DSA is similar to the ElGamal signature scheme.

2.1 Supported Algorithms

The available digital signature algorithms are listed below:

1. **RSA**

RSA is public key cryptosystem designed by Ronald Rivest, Adi Shamir and Leonard Adleman. It can be used with any hash functions.

2. **DSA**

DSA is the USA's Digital Signature Standard. It uses only the SHA-1 hash algorithm.
The supported cryptographic hash algorithms are:

3. **MD2**

MD2 is a cryptographic hash algorithm designed by Ron Rivest. It is optimized for 8-bit processors. Outputs 128 bits of data. There are no known weaknesses of this algorithm but since this algorithm is rarely used and not really studied it should not be used today.

4. **MD5**

MD5 is a cryptographic hash algorithm designed by Ron Rivest. Outputs 128 bits of data. It is considered to be broken.

5. **SHA-1**

SHA is a cryptographic hash algorithm designed by NSA. Outputs 160 bits of data. It is also considered to be broken, though no practical attacks have been found.

6. **RMD160**

RIPEMD is a cryptographic hash algorithm developed in the framework of the EU project RIPE. Outputs 160 bits of data.

2.2Correctness of an Algorithm:

The signature scheme is correct in the sense that the verifier will always accept genuine signatures.

This can be shown as follows:

First, if $g = h^{(p-1)/q}$ mod p it follows that $g^q \equiv h^{p-1} \equiv 1$ (mod p) by Fermat's little theorem. Since $g>1$ and q is prime, g must have order q.
The signer computes

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

Thus

$$\begin{aligned} k &\equiv H(m)s^{-1} + xrs^{-1} \\ &\equiv H(m)w + xrw \pmod q. \end{aligned}$$

Since g has order q we have

$$\begin{aligned} g^k &\equiv g^{H(m)w} g^{xrw} \\ &\equiv g^{H(m)w} y^{rw} \\ &\equiv g^{u1} y^{u2} \pmod p. \end{aligned}$$

Finally, the correctness of DSA follows from

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

3. Use of the DSA algorithm

The DSA 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 both signature generation and verification, the data which is referred to as a message, M, is reduced by means of the Secure Hash Algorithm (SHA) specified in FIPS YY. 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. This binding may be certified by a mutually trusted party. For example, a certifying authority could sign credentials containing a user's public key and identity to form a certificate. Systems for certifying credentials and distributing certificates are beyond the scope of this standard. NIST intends to publish separate document(s) on certifying credentials and distributing certificates.

4. DSA parameters

The DSA makes use of the following parameters:

1. p = a prime modulus, where $2^{L-1} < p < 2^L$ for 512 = < L = <1024 and L a multiple of 64
2. q = a prime divisor of p - 1, where $2^{159} < q < 2^{160}$
3. $g = h^{(p-1)/q} \mod p$, where h is any integer with $1 < h < p - 1$ such that $h^{(p-1)/q} \mod p > 1$ (g has order q mod p)
4. x = a randomly or pseudo randomly generated integer with $0 < x < q$
5. $y = g^x \mod p$
6. k = a randomly or pseudo randomly generated integer with $0 < k < q$

The integer's p, q, and g can be public and can be common to a group of users. A user's private and public keys are x and y, respectively. They are normally fixed for a period of time. Parameters x and k are used for signature generation only, and must be kept secret. Parameter k must be regenerated for each signature. Parameters p and q shall be generated as specified in Appendix 2, or using other FIPS approved security methods. Parameters x and k shall be generated as specified in Appendix 3, or using other FIPS approved security methods.

5. Signature generation

The signature of a message M is the pair of numbers r and s computed according to the equations below:

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

$s = (k^{-1}(SHA(M) + xr)) \mod q$.

In the above, k^{-1} is the multiplicative inverse of k, mod q; i.e., $(k^{-1} k) \mod q = 1$ and $0 < k^{-1} < q$. The value of SHA(M) is a 160-bit string output by the Secure Hash Algorithm specified in FIPS 180. For use in computing s, this string must be converted to an integer.

As an option, one may wish to check if r = 0 or s = 0. If either r = 0 or s = 0, a new value of k should be generated and the signature should be recalculated (it is extremely unlikely that r = 0 or s = 0 if signatures are generated properly). The signature is transmitted along with the message to the verifier.

6. Signature verification

Prior to verifying the signature in a signed message, p, q and g plus the sender's public key and identity are made available to the verifier in an authenticated manner.

Let M', r' and s' be the received versions of M, r, and s, respectively, and let y be the public key of the signatory. To verifier first checks to see that $0 < r' < q$ and $0 < s' < q$; if either condition is violated the signature shall be rejected. If these two conditions are satisfied, the verifier computes

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

$u1 = ((SHA(M')w) \bmod q$

$u2 = ((r')w) \bmod q$

$v = (((g)^{ul} (y)^{u2}) \bmod p) \bmod q.$

If v = r', then the signature is verified and the verifier can have high confidence that the received message was sent by the party holding the secret key x corresponding to y. For a proof that v = r' when M' = M, r' = r, and s' = s, see Appendix1. If v does not equal r', then the message may have been modified, the message may have been incorrectly signed by the signatory, or the message may have been signed by an impostor. The message should be considered invalid.

7. Conclusion

Many security technologies use digital signatures. For example, Microsoft® Authenticode® can be used to digitally sign software programs, safeguarding them when they are distributed on the intranet or Internet to help counter the threat of software tampering and the spread of viruses and other malicious code. Likewise, the S/MIME protocol can be used to digitally sign e-mail messages to ensure the integrity of mail communications.

8. References

1. William Stallings, “Cryptography and Network Security: Principles and Practice”, 3rd edition, Prentice Hall, 2003.
2. Bidzos, Jim, "Threats to Privacy and Public Keys for Protection", *COMPCON Spring '91 Digest of* Papers, IEEE Computer Society Press, p. 189-94
3. Singhal, Mukesh and Shivaratri, Niranjan G., Advanced Concepts in Operating Systems, McGraw-Hill, p. 405 8.Rivest, R.L., Shamir, A., and Adleman, L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Communications of the ACM*, Vol 21, No. 2, February 1978, p. 120-26
4. D. Boneh. \Twenty Years of Attacks on the RSA Cryptosystem." Notices of the American Mathematical Society, 46(2):203{213, Feb.1999.
5. D. Boneh and G. Durfee. \Cryptanalysis of RSA with Private Key d Less than n0:292." IEEE Transactions on Information Theory, 46(4):1339{1349, Jul. 2000. Early version in Proceedings of Eurocrypt '99.
6. M. Wiener. \Cryptanalysis of Short RSA Secret Exponents." IEEE Trans. on Info. Th. 36(3):553 {558. May 1990.
7. R. Rivest, A. Shamir and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, 21 (2), pp. 120-126, February 1978.
8. RSA Security, Inc., “Public-Key Cryptography Standards”,