WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Català
Čeština
Deutsch
Español
فارسی
Français
Italiano
עברית
Lietuvių
日本語
Polski
Português
Română
Русский
Suomi
Türkçe
Tiếng Việt
Edit links

Digital Signature Algorithm

From Wikipedia, the free encyclopedia

The **Digital Signature Algorithm** (**DSA**) is a Federal Information Processing Standard 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**) and adopted as FIPS 186 in 1993.[1] Four revisions to the initial specification have been released: FIPS 186-1 in 1996,[2] FIPS 186-2 in 2000,[3] FIPS 186-3 in 2009,[4] and FIPS 186-4 in 2013.[5]

DSA is covered by U.S. Patent 5,231,668, filed July 26, 1991 and attributed to David W. Kravitz,[6] a former NSA employee. This patent was given to "The United States of America as represented by the Secretary of Commerce, Washington, D.C.", and NIST has made this patent available worldwide royalty-free.[7] Claus P. Schnorr claims that his U.S. Patent 4,995,082 (expired) covered DSA; this claim is disputed.[8] DSA is a variant of the ElGamal Signature Scheme.

Contents [hide]

Key generation [edit]

Key generation has two phases. The first phase is a choice of *algorithm parameters* which may be shared between different users of the system, while the second phase computes public and private keys for a single user.

Parameter generation [edit]

- Choose an approved cryptographic hash function H. In the original DSS, H was always SHA-1, but the stronger SHA-2 hash functions are approved for use in the current DSS.[5][9] The hash output may be truncated to the size of a key pair.
- Decide on a key length L and N. 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 (inclusive). NIST 800-57 recommends lengths of 2048 (or 3072) for keys with security lifetimes extending beyond 2010 (or 2030), using correspondingly longer N.[10] FIPS 186-3 specifies L and N length pairs of (1024,160), (2048,224), (2048,256), and (3072,256).[4]
- Choose an N-bit prime q. N must be less than or equal to the hash output length.
- Choose an L-bit prime modulus p such that 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 with a different h if the result comes out as 1. Most choices of h will lead to a usable g; commonly h=2 is used.

The algorithm parameters (p, q, g) may be shared between different users of the system.

Per-user keys [edit]

Given a set of parameters, 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.

There exist efficient algorithms for computing the modular exponentiations $h^{(p–1)/q}$ mod p and g^x mod p, such as exponentiation by squaring.

Signing [edit]

Let H be the hashing function and m the message:

- Generate a random per-message value k where $0 < k < q$
- Calculate $r = \left(g^k \bmod p\right) \bmod q$
- In the unlikely case that $r = 0$, start again with a different random k
- Calculate $s = k^{-1}\left(H\left(m\right) + xr\right) \bmod q$
- In the unlikely case that $s = 0$, start again with a different random k
- The signature is $\left(r, s\right)$

The first two steps amount to creating a new per-message key. The modular exponentiation here is the most computationally expensive part of the signing operation, and it may be computed before the message hash is known. The modular inverse $k^{-1} \bmod q$ is the second most expensive part, and it may also be computed before the message hash is known. It may be computed using the extended Euclidean algorithm or using Fermat's little theorem as $k^{q-2} \bmod q$.

Verifying [edit]

- Reject the signature if $0 < r < q$ or $0 < s < q$ is not satisfied.
- Calculate $w = s^{-1} \bmod q$
- Calculate $u_1 = H\left(m\right) \cdot w \bmod q$
- Calculate $u_2 = r \cdot w \bmod q$
- Calculate $v = \left(\left(g^{u_1} y^{u_2}\right) \bmod p\right) \bmod q$
- The signature is invalid unless $v = r$

DSA is similar to the ElGamal signature scheme.

Correctness of the algorithm [edit]

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

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

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 (mod p) 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

$$\begin{aligned} r &= (g^k \bmod p) \bmod q \\ &= (g^{u1} y^{u2} \bmod p) \bmod q \\ &= v \end{aligned}$$

Sensitivity [edit]

With DSA, the entropy, secrecy, and uniqueness of the random signature value k is critical. It is so critical that violating any one of those three requirements can reveal the entire private key to an attacker.[11] Using the same value twice (even while keeping k secret), using a predictable value, or leaking even a few bits of k in each of several signatures, is enough to break DSA.[12]

This issue affects both DSA and ECDSA - in December 2010, a group calling itself *fail0verflow* announced recovery of the ECDSA private key used by Sony to sign software for the PlayStation 3 game console. The attack was made possible because Sony failed to generate a new random *k* for each signature.[13]

This issue can be prevented by deriving *k* deterministically from the private key and the message hash, as described by RFC 6979. This ensures that *k* is different for each *H(m)* and unpredictable for attackers who do not know the private key *x*.

See also [edit]

- Elliptic Curve Digital Signature Algorithm
- Modular arithmetic

References [edit]

1. ^ "FIPS PUB 186]: Digital Signature Standard (DSS), 1994-05-19". *csrc.nist.gov*.
2. ^ "FIPS PUB 186-1: Digital Signature Standard (DSS), 1998-12-15" (PDF). *csrc.nist.gov*.
3. ^ "FIPS PUB 186-2: Digital Signature Standard (DSS), 2000-01-27" (PDF). *csrc.nist.gov*.
4. ^ *a* *b* "FIPS PUB 186-3: Digital Signature Standard (DSS), June 2009" (PDF). *csrc.nist.gov*.
5. ^ *a* *b* "FIPS PUB 186-4: Digital Signature Standard (DSS), July 2013" (PDF). *csrc.nist.gov*.
6. ^ Dr. David W. Kravitz
7. ^ Werner Koch. DSA and patents
8. ^ Minutes of the Sept. 94 meeting of the Computer System Security and Privacy Advisory Board
9. ^ "FIPS PUB 180-4: Secure Hash Standard (SHS), March 2012" (PDF). *csrc.nist.gov*.
10. ^ "NIST Special Publication 800-57" (PDF). *csrc.nist.gov*.
11. ^ "The Debian PGP disaster that almost was". *root labs rdist*.
12. ^ DSA *k*-value Requirements
13. ^ Bendel, Mike (2010-12-29). "Hackers Describe PS3 Security As Epic Fail, Gain Unrestricted Access". Exophase.com. Retrieved 2011-01-05.

External links [edit]

- FIPS PUB 186-4: Digital Signature Standard (DSS), the fourth (and current) revision of the official DSA specification.

- Recommendation for Key Management -- Part 1: general, NIST Special Publication 800-57, p. 62–63

v · t · e	**Public-key cryptography**
Algorithms	AEDH · Benaloh · Blum–Goldwasser · Cayley–Purser · CEILIDH · Cramer–Shoup · Damgård–Jurik · DH · **DSA** · EPOC · ECDH · ECDSA · EdDSA · EKE · ElGamal (signature scheme) · GMR · Goldwasser–Micali · HFE · IES · Lamport · McEliece · Merkle–Hellman · MQV · Naccache–Stern · Naccache–Stern knapsack cryptosystem · NTRUEncrypt · NTRUSign · Paillier · Rabin · RSA · Okamoto–Uchiyama · Schnorr · Schmidt–Samoa · SPEKE · SRP · STS · Three-pass protocol · XTR
Theory	Discrete logarithm · Elliptic curve cryptography · Non-commutative cryptography · RSA problem
Standardization	CRYPTREC · IEEE P1363 · NESSIE · NSA Suite B
Topics	Digital signature · OAEP · Fingerprint · PKI · Web of trust · Key size

v · t · e	**Cryptography**
	History of cryptography · Cryptanalysis · Cryptography portal · Outline of cryptography
	Symmetric-key algorithm · Block cipher · Stream cipher · Public-key cryptography · Cryptographic hash function · Message authentication code · Random numbers · Steganography

Categories: Digital signature schemes

This page was last modified on 6 August 2015, at 10:13.