ElGamal signature scheme

The **ElGamal signature scheme** is a digital signature scheme which is based on the difficulty of computing discrete logarithms. It was described by Taher ElGamal in 1984.[1]

The ElGamal signature algorithm described in this article is rarely used in practice. A variant developed at NSA and known as the Digital Signature Algorithm is much more widely used. There are several other variants.[2] The ElGamal signature scheme must not be confused with ElGamal encryption which was also invented by Taher ElGamal.

The ElGamal signature scheme allows that a verifier can confirm the authenticity of a message *m* sent by the signer sent to him over an insecure channel.

System parameters

- Let *H* be a collision-resistant hash function.
- Let *p* be a large prime such that computing discrete logarithms modulo *p* is difficult.
- Let $g < p$ be a randomly chosen generator of the multiplicative group of integers modulo *p* Z_p^*.

These system parameters may be shared between users.

Key generation

- Choose randomly a secret key *x* with $1 < x < p - 1$.
- Compute $y = g^x \bmod p$.
- The public key is (p, g, y).
- The secret key is *x*.

These steps are performed once by the signer.

Signature generation

To sign a message *m* the signer performs the following steps.

- Choose a random *k* such that $0 < k < p - 1$ and $\gcd(k, p - 1) = 1$.
- Compute $r \equiv g^k \pmod p$.
- Compute $s \equiv (H(m) - xr)k^{-1} \pmod{p-1}$.
- If $s = 0$ start over again.

Then the pair (r,s) is the digital signature of *m*. The signer repeats these steps for every signature.

Verification

A signature (r,s) of a message *m* is verified as follows.

- $0 < r < p$ and $0 < s < p - 1$.
- $g^{H(m)} \equiv y^r r^s \pmod p$.

The verifier accepts a signature if all conditions are satisfied and rejects it otherwise.

Correctness

The algorithm is correct in the sense that a signature generated with the signing algorithm will always be accepted by the verifier.

The signature generation implies

$$H(m) \equiv xr + sk \pmod{p-1}.$$

Hence Fermat's little theorem implies

$$\begin{aligned} g^{H(m)} &\equiv g^{xr} g^{ks} \\ &\equiv (g^x)^r (g^k)^s \\ &\equiv (y)^r (r)^s \pmod{p}. \end{aligned}$$

Security

A third party can forge signatures either by finding the signer's secret key *x* or by finding collisions in the hash function $H(m) \equiv H(M) \pmod{p-1}$. Both problems are believed to be difficult. However, as of 2011 no tight reduction to a computational hardness assumption is known.

The signer must be careful to choose a different *k* uniformly at random for each signature and to be certain that *k*, or even partial information about *k*, is not leaked. Otherwise, an attacker may be able to deduce the secret key *x* with reduced difficulty, perhaps enough to allow a practical attack. In particular, if two messages are sent using the same value of *k* and the same key, then an attacker can compute *x* directly.[1]

References

[1] T. ElGamal (1985). "A public key cryptosystem and a signature scheme based on discrete logarithms" (http://hereford.homeip.net/ElGamal/ElGamal - A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms ElGamal.pdf). *IEEE Trans inf Theo* **31** (4): 469–472. .

[2] K. Nyberg, R. A. Rueppel (1996). "Message recovery for signature schemes based on the discrete logarithm problem" (http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/E94/182.PDF). *Designs, Codes and Cryptography* **7** (1-2): 61–81. doi:10.1007/BF00125076. .

Article Sources and Contributors

ElGamal signature scheme *Source*: http://en.wikipedia.org/w/index.php?oldid=412263425 *Contributors*: ArnoldReinhold, Bbartlog, Ciphergoth, Davidgothberg, Fryed-peach, Gelbard, Intgr, Jafet, John Reaves, Jopsen, JustAGal, Kevinsluckynumbersev3n, Matt Crypto, Maxal, Message From Xenu, Michael Hardy, Reetep, Senojsitruc, Ww, 27 anonymous edits

License