PBKDF2

PBKDF2 (**Password-Based Key Derivation Function 2**) is a key derivation function that is part of RSA Laboratories' Public-Key Cryptography Standards (PKCS) series, specifically PKCS #5 v2.0, also published as Internet Engineering Task Force's RFC 2898. It replaces an earlier standard, **PBKDF1**, which could only produce derived keys up to 160 bits long.

PBKDF2 applies a pseudorandom function, such as a cryptographic hash, cipher, or HMAC to the input password or passphrase along with a salt value and repeats the process many times to produce a *derived key*, which can then be used as a cryptographic key in subsequent operations. The added computational work makes password cracking much more difficult, and is known as key stretching. When the standard was written in 2000, the recommended minimum number of iterations was 1000, but the parameter is intended to be increased over time as CPU speeds increase. Having a salt added to the password reduces the ability to use precomputed hashes (rainbow tables) for attacks, and means that multiple passwords have to be tested individually, not all at once. The standard recommends a salt length of at least 64 bits.

Key derivation process

The PBKDF2 key derivation function has five input parameters:

```
DK = PBKDF2(PRF, Password, Salt, c, dkLen)
```

where:

- *PRF* is a pseudorandom function of two parameters with output length *hLen* (e.g. a keyed HMAC)
- *Password* is the master password from which a derived key is generated
- *Salt* is a cryptographic salt
- *c* is the number of iterations desired
- *dkLen* is the desired length of the derived key
- *DK* is the generated derived key

Each *hLen*-bit block T_i of derived key DK, is computed as follows:

```
DK = T_1 || T_2 || ... || T_dklen/hlen
T_i = F(Password, Salt, c, i)
```

The function *F* is the xor (^) of *c* iterations of chained PRFs. The first iteration of PRF uses *Password* as the PRF key and *Salt* concatenated with *i* encoded as a big-endian 32-bit integer. (Note that *i* is a 1-based index.) Subsequent iterations of PRF use *Password* as the PRF key and the output of the previous PRF computation as the salt:

```
F(Password, Salt, c, i) = U_1 ^ U_2 ^ ... ^ U_c
```

where:

```
U_1 = PRF(Password, Salt || INT_32_BE(i))
U_2 = PRF(Password, U_1)
...
U_c = PRF(Password, U_c-1)
```

For example, WPA2 uses:

```
 DK = PBKDF2(HMAC-SHA1, passphrase, ssid, 4096, 256)
```

Implementations

- openssl's C implementation [1]
- OpenBSD's C implementation [2]
- PolarSSL's C implementation [3]
- CyaSSL's C implementation [4]
- ActionScript 3.0 implementation [5]
- .NET's built-in function [6]
- C# implementation [7]
- Go implementation [8]
- JavaScript implementations slow [9], less slow [10], fast [11], benchmark [12]
- Java implementation (PBKDF2WithHmacSHA1) [13]
- Python implementation [14]
- Perl implementation [15]
- Ruby's standard library [16]
- Ruby implementation [17]
- REBOL2 implementation [18]
- PHP implementations: native [19] (added in v5.5.0), pure PHP implementation [20]
- Scala implementation [21]

Systems that use PBKDF2

- Wi-Fi Protected Access (WPA and WPA2) used to secure Wi-Fi wireless networks
- Microsoft Windows Data Protection API (DPAPI)
- OpenDocument encryption used in OpenOffice.org
- WinZip's AES Encryption scheme.
- LastPass for password hashing.
- Dashlane for password hashing.
- Apple's iOS mobile operating system, for protecting user passcodes and passwords.[22]
- Mac OS X Mountain Lion for user passwords[*citation needed*]
- The Django web framework, as of release 1.4.
- The MODX content management framework, as of version 2.0.
- The encryption and decryption schema of Zend Framework, to generate encryption and authentication keys.[23]
- Cisco IOS and IOS XE Type 4 password hashes

Disk encryption software

- Filesystem encryption in the Android operating system, as of version 3.0.[24]
- FileVault (Mac OS X) from Apple Computer[25]
- FreeOTFE (Windows and Pocket PC PDAs); also supports mounting Linux (e.g. LUKS) volumes under Windows
- LUKS (Linux Unified Key Setup) (Linux)
- TrueCrypt (Windows, Linux, and Mac OS X)
- DiskCryptor (Windows)
- Cryptographic disk [26] (NetBSD)
- GEOM ELI module for FreeBSD
- softraid [27] crypto for OpenBSD
- EncFS (Linux, FreeBSD and Mac OS X) since v1.5.0
- GRUB2 (boot loader)

Alternatives to PBKDF2

One weakness of PBKDF2 is that while its number of iterations can be adjusted to make it take an arbitrarily large amount of computing time, it can be implemented with a small circuit and very little RAM, which makes brute-force attacks using ASICs or GPUs relatively cheap.[28] The bcrypt key derivation function requires a larger (but still fixed) amount of RAM and is slightly stronger against such attacks, while the more modern scrypt key derivation function can use arbitrarily large amounts of memory and is therefore more resistant to ASIC and GPU attacks.

References

[1] http://cvs.openssl.org/rlog?f=openssl/crypto/evp/p5_crpt2.c
[2] http://www.openbsd.org/cgi-bin/cvsweb/src/lib/libutil/pkcs5_pbkdf2.c?rev=HEAD
[3] https://polarssl.org/pbkdf2-source-code
[4] http://www.yassl.com/yaSSL/Source/output/ctaocrypt/src/pwdbased.c.html
[5] http://code.google.com/p/as3-pbkdf2/
[6] http://msdn.microsoft.com/en-us/library/system.security.cryptography.rfc2898derivebytes.aspx
[7] http://msdn.microsoft.com/en-us/magazine/cc163913.aspx
[8] https://code.google.com/p/go/source/browse/pbkdf2/pbkdf2.go?repo=crypto
[9] http://anandam.name/pbkdf2
[10] http://code.google.com/p/crypto-js/
[11] http://bitwiseshiftleft.github.com/sjcl/
[12] http://jsperf.com/pbkdf2-sjcl-cryptojs
[13] http://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html#SecretKeyFactory
[14] http://www.dlitz.net/software/python-pbkdf2
[15] https://metacpan.org/module/Crypt::PBKDF2
[16] http://www.ruby-doc.org/stdlib-1.9.3/libdoc/openssl/rdoc/OpenSSL/PKCS5.html#method-c-pbkdf2_hmac_sha1
[17] http://github.com/emerose/pbkdf2-ruby/tree/master
[18] https://github.com/refaktor/Rebol2-PBKDF2
[19] http://php.net/hash-pbkdf2
[20] https://defuse.ca/php-pbkdf2.htm
[21] http://github.com/nremond/pbkdf2-scala
[22] iOS security (http://images.apple.com/ipad/business/docs/iOS_Security_May12.pdf), May 2012, Apple inc.
[23] Encrypt/decrypt using block ciphers (http://zf2.readthedocs.org/en/latest/modules/zend.crypt.block-cipher.html), Programmer's Reference Guide of Zend Framework 2.
[24] Notes on the implementation of encryption in Android 3.0 (http://source.android.com/tech/encryption/android_crypto_implementation.html), September 2012, Android Open Source Project.
[25] http://crypto.nsa.org/vilefault/23C3-VileFault.pdf
[26] http://netbsd.org/guide/en/chap-cgd.html
[27] http://www.openbsd.org/cgi-bin/man.cgi?query=softraid&sektion=4&apropos=0&manpath=OpenBSD+Current&arch=i386
[28] Colin Percival. scrypt (http://www.tarsnap.com/scrypt.html). As presented in "Stronger Key Derivation via Sequential Memory-Hard Functions" (http://www.tarsnap.com/scrypt/scrypt.pdf). presented at BSDCan'09, May 2009.

External links

- RSA PKCS #5 (http://www.rsa.com/rsalabs/node.asp?id=2127) – RSA Laboratories PKCS #5 v2.0 - Multiple Formats, and test vectors.
- RFC 2898 – Specification of PKCS #5 v2.0.
- RFC 6070 – Test vectors for PBKDF2 with HMAC-SHA1.
- NIST Special Publication 800-132 Recommendation for Password-Based Key Derivation (http://csrc.nist.gov/publications/nistpubs/800-132/nist-sp800-132.pdf)

Article Sources and Contributors

PBKDF2 *Source*: http://en.wikipedia.org/w/index.php?oldid=599625763 *Contributors*: A5b, Abadger2k6, Ae.envy, ArnoldReinhold, Astralblue, Billymeltdown, Brainiac86, Breadtk, Chris conlon, Cperciva, CraSH, Cralar, Cstrep, Daira Hopwood, DavidCary, DirkHelgemo, DouglasCalvert, Download, Emlynoregan, Gmadey, Gogo Dodo, GoingBatty, GregorB, Ilmari Karonen, Intgr, JPGoldberg, Jas4711, Jettlogic, Korin43, Kslays, Leotohill, Liambowen, Maartenvanrooijen, Matt Crypto, Mattghali, MidnightLightning, Mr.Unknown, Muelleum, Nageh, Nealmcb, Nn123645, Nremond, ORBIT, Ondertitel, OneSafe, PMatthew, Pavritch, PeterWesco, Piet Delport, Plazmatyk, Quelrod, Raymond Hill, Securiger, Senator2029, Shadowjams, Sjlombardo, Softtest123, Strongriley, Sweerek, TRauMa, Tbvdm, Teratorn, Tgr, Turtle595, Two Bananas, Typefreak, Weichholdo, Wikisian, Wonderstruck, Wrs1864, Xpclient, 78 anonymous edits

License