Kerberos Working Group
Updates: Kerberos-revisions
Document: draft-raeburn-krb-rijndael-krb-01.txt

K. Raeburn
MIT

July 2, 2001

Rijndael, Serpent, and Twofish Cryptosystems for Kerberos 5

Status of this Memo

1. Abstract

The AES competition in the US [AES] has prompted the submission and analysis of a number of new ciphers intended to be significantly stronger and faster than the old DES algorithm. This document describes the addition of some of these algorithms to the Kerberos cryptosystem suite [Kerb].

Comments should be sent to the author, or to the IETF Kerberos working group (ietf-krb-wg@anl.gov).

2. Conventions Used in this Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

3. New Encryption and Checksum Types

This document defines encryption key and checksum types for Kerberos 5 to be used with the Rijndael (chosen by NIST as the AES cipher), Twofish and Serpent encryption algorithms.

Each of these algorithms, as required by the AES specifications, supports 128-bit block encryption, and key sizes of 128, 192, or 256 bits. Other block sizes and key sizes are also supported by some of these algorithms, but are not considered here.

Using the "simplified profile" of section 6 of [Kerb], we can define a group of encryption and checksum schemes. The basic encryption algorithms listed above are used in CBC mode, with a zero initial vector. The associated checksum function is the function from [SHA256] with an output size of twice the key size (SHA256, SHA384, or SHA512).

We use the above encryption algorithms in CBC mode, and a hash algorithm of SHA256, SHA384, or SHA512 [SHA256], with the hash size twice the size of the encryption key, in the "simplified profile" of section 6 of [Kerb]. (As of the time of this writing, NIST is reviewing several new proposed modes of operation, some of which may permit encryption and integrity protection simultaneously. Once this process is done, we may wish to consider using one of these modes to define a new profile.) Unless otherwise specified, a zero initial vector must be used for CBC mode.

4. Key Generation From Pass Phrases

For each of these new encryption/checksum profiles, we define a process for generating a key from a pass phrase and salt string, both assumed to be supplied in UTF-8 representation.

We use the PBKDF2 function from PKCS #5 v2.0 ([RFC2898]), with parameters indicated below, to generate an intermediate key, which is passed into the key derivation algorithm with the constant string "kerberos" as in [Kerb]. The resulting key is the user's long-term key for use with the encryption algorithm in question:

```
tkey = PBKDF2 (passphrase, salt, 2, keylength)
key = DK(tkey, "kerberos")
```

(As defined, PBKDF2 actually generates a random byte string, and the PKCS #5 spec assumes that the key space is dense. For the pedantic, we get a "random byte string" from PBKDF2, and pass it through the random-to-key function in the profile -- which we define as a simple copy operation -- to get a "key".)

The pseudorandom function used by PBKDF2 will be an HMAC of the passphrase and salt, as described in Appendix B.1 to PKCS#5, but using the hash function chosen for the encryption algorithm instead of SHA-1. For example, all of the 192-bit-key profiles will use an HMAC based on SHA384.

Note that since the hash function's output block size is larger than the key sizes, only one block needs to be generated through the PBKDF2 algorithm, and only two iterations are specified. Thus, using the notation from PKCS#5, the intermediate key is given by:

```
U_1 = PRF(P, S || INT(0))
U_2 = PRF(P, U_1)
tmp = U_1 \xor U_2
tkey = tmp<0..keylength-1>
```

Sample test vectors are given in the appendix.

5. Kerberos Algorithm Profile Parameters

This is a summary of the parameters to be used with the simplified algorithm profile described in section 6.4.2 of [Kerb]:

protocol key format	simple byte string of 128, 192 or 256 bits
string-to-key function	PBKDF2+DK (see previous section)
key-generation seed length	key size
random-to-key function	identity
hash function	SHA-256, -384 or -512 with output size twice the key size
block size	128 bits
encryption and	Rijndael, Serpent or

decryption functions	Twofish, in CBC mode with zero ivec

Expanding on the set of key sizes and the set of encryption functions, this gives us nine profiles for encryption and checksum algorithm pairs.

6. Assigned Numbers (Cliff? IANA?)

The following encryption type numbers are assigned:

encryption types		
type name	etype value	key size
rijndael128-hmac-sha256	TBD	128
rijndael192-hmac-sha384	TBD	192
rijndael256-hmac-sha512	TBD	256
serpent128-hmac-sha256	TBD	128
serpent192-hmac-sha384	TBD	192
serpent256-hmac-sha512	TBD	256
twofish128-hmac-sha256	TBD	128
twofish192-hmac-sha384	TBD	192
twofish256-hmac-sha512	TBD	256

Where implementations accept type names in human-readable form, the alternative names "aes128-hmac-sha256", "aes192-hmac-sha384" and "aes256-hmac-sha512" are recommended to be permitted as aliases for the Rijndael key types.

The following checksum type numbers are assigned:

checksum types		
type name	sumtype value	length

	hmac-sha256-rijndael128	TBD	256
	hmac-sha384-rijndael192	TBD	384
	hmac-sha512-rijndael256	TBD	512
	hmac-sha256-serpent128	TBD	256
	hmac-sha384-serpent192	TBD	384
	hmac-sha512-serpent256	TBD	512
	hmac-sha256-twofish128	TBD	256
	hmac-sha384-twofish192	TBD	384
	hmac-sha512-twofish256	TBD	512

These checksum types will be used with the corresponding encryption types defined above. Aliases "hmac-sha256-aes128" and so forth are suggested for the checksum types associated with Rijndael keys.

7. Recommendations

Rijndael, as the proposed AES cipher, is strongly RECOMMENDED, with all three lengths.

Twofish and Serpent, described in the AES report as weaker that Rijndael in terms of performance or implementability in certain environments but stronger in terms of anticipated resistance to certain types of possible attacks, are OPTIONAL.

8. Security Considerations

These new algorithms have not been around long enough to receive the decades of intense analysis that DES has received. It is possible that some weakness exists that has not been found by the cryptosystems' authors or other cryptographers analyzing these algorithms before and during the AES competition. The AES report does indicate that arguments were put forth relating to this in favor of deploying multiple algorithms in case one is found to be significantly weaker than previously believed.

The 256-bit SHA algorithm is a work in progress by the US National Institute of Standards and Technology. To the best of the author's knowledge, the review process has not been completed. The use of this algorithm in this document is with the assumption that the standardization process will go smoothly.

The author is not a cryptographer.

9. Acknowledgements

Thanks to John Brezak for feedback on earlier versions of this document.

10. References

[AES] Nechvatal, J., Barker, E., Bassham, L., Burr, W., Dworkin, M., Foti, J., Roback, E., "Report on the Development of the Advanced Encryption Standard (AES)", National Institute of Standards and Technology, October 2, 2000.

[Kerb] Neuman, C., Kohl, J., Ts'o, T., Raeburn, K., Yu, T., "The Kerberos Network Authentication Service (V5)", draft-ietf-cat-kerberos-revisions-08.txt, March 2, 2001. Work in progress.

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", RFC 2026, October, 1996.

[RFC2898] Kaliski, B., "PKCS #5: Passowrd-Based Cryptography Specification Version 2.0", RFC 2898, September 2000.

[Rijn] Daemen, J., Rijmen, V., "AES Proposal: Rijndael", September 3, 1999. *

[Serp] Anderson, R., Biham, E., Knudsen, L., "Serpent: A Proposal for the Advanced Encryption Standard", June 1998. *

[SHA256] NIST doc ... *

[Twof] Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., Ferguson, N., "The Twofish Encrytion Algorithm: A 128-Bit Block Cipher", Wiley Computer Publishing, 1999.

* Need more substantial references (RFCs or published papers) if possible; web-accessible copy may not be a permanent reference.

11. Author's Address

Kenneth Raeburn

Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139
raeburn@mit.edu

12. Full Copyright Statement

A. Sample test vectors

Some sample test vectors for the string-to-key algorithm:

(values to be filled in later)

Pass phrase: "test"
Salt: none

```
  74 65 73 74
PBKDF2 128-bit output (intermediate key):
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
Rijndael 128-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
Serpent 128-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
Twofish 128-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
PBKDF2 192-bit output (intermediate key):
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx
Rijndael 192-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx
Serpent 192-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx
Twofish 192-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx
PBKDF2 256-bit output (intermediate key):
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
Rijndael 256-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
Serpent 256-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
Twofish 256-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
```


Pass phrase: "password"

```
  70 61 73 73 77 6f 72 64
Salt: "ATHENA.MIT.EDUraeburn"
  41 54 48 45 4e 41 2e 4d 49 54 2e 45 44 55 72 61
  65 62 75 72 6e
PBKDF2 128-bit output (intermediate key):
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
Rijndael 128-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
Serpent 128-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
Twofish 128-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
PBKDF2 192-bit output (intermediate key):
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx
Rijndael 192-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx
Serpent 192-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx
Twofish 192-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx
PBKDF2 256-bit output (intermediate key):
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
Rijndael 256-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
Serpent 256-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
Twofish 256-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx

Pass phrase: eszett
  c3 9f
Salt: "ATHENA.MIT.EDUJuri" + s-caron + "i" + c-acute
  41 54 48 45 4e 41 2e 4d 49 54 2e 45 44 55 4a 75
  72 69 c5 a1 69 c4 87
PBKDF2 128-bit output (intermediate key):
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
Rijndael 128-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
```

```
Serpent 128-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
Twofish 128-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
PBKDF2 192-bit output (intermediate key):
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx
Rijndael 192-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx
Serpent 192-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx
Twofish 192-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx
PBKDF2 256-bit output (intermediate key):
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
Rijndael 256-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
Serpent 256-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
Twofish 256-bit key:
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
```

B. Change History

Delete this section before RFC publication.

Major changes from -00:

Define different types based on key/hash sizes, with hash size always twice key size. Use simplified profile of revised section 6 of RFC1510bis. Drop "-kd" from the names.

Use PKCS#5 instead of simple hash. Changed string-to-key vector to use some "Appendix Z" cases also submitted for kerberos-revisions.