INTERNET DRAFT
draft-ietf-otp-01.txt
March 24, 1997

Neil Haller
Bellcore
Craig Metz
Kaman Sciences Corporation
Philip Nesser
Nesser & Nesser Consulting
Mike Straw
Bellcore

A One-Time Password System

STATUS OF THIS MEMO

1.0 ABSTRACT

This document describes a one-time password authentication system (OTP). The system provides authentication for system access (login) and other applications requiring authentication that is secure against passive attacks based on replaying captured reusable passwords. OTP evolved from the S/KEY* One-Time Password System that was released by Bellcore and is described in references [3] and [5].

2.0 OVERVIEW

One form of attack on networked computing systems is eavesdropping on network connections to obtain authentication information such as the login IDs and passwords of legitimate users. Once this information is captured, it can be used at a later time to gain

* S/KEY is a trademark of Bellcore

to the system. One-time password systems are designed to counter this type of attack, called a "replay attack" [4].

The authentication system described in this document uses a secret pass-phrase to generate a sequence of one-time (single use) passwords. With this system, the user's secret pass-phrase never needs to cross the network at any time such as during authentication or during pass-phrase changes. Thus, it is not vulnerable to replay attacks. Added security is provided by the property that no secret information need be stored on any system, including the server being protected.

The OTP system protects against external passive attacks against the authentication subsystem. It does not prevent a network eavesdropper from gaining access to private information and does not provide protection against either "social engineering" or active attacks [9].

3.0 INTRODUCTION

There are two entities in the operation of the OTP one-time password system. The generator must produce the appropriate one-time password from the user's secret pass-phrase and from information provided in the challenge from the server. The server must send a challenge that includes the appropriate generation parameters to the generator, must verify the one-time password received, must store the last valid one-time password it received, and must store the corresponding one-time password sequence number. The server must also facilitate the changing of the user's secret pass-phrase in a secure manner.

The OTP system generator passes the user's secret pass-phrase, along with a seed received from the server as part of the challenge, through multiple iterations of a secure hash function to produce a one-time password. After each successful authentication, the number of secure hash function iterations is reduced by one. Thus, a unique sequence of passwords is generated. The server verifies the one-time password received from the generator by computing the secure hash function once and comparing the result with the previously accepted one-time password. This technique was first suggested by Leslie Lamport [1].

4.0 REQUIREMENTS TERMINOLOGY

In this document, the words that are used to define the significance of each particular requirement are usually capitalized. These words are:

- MUST

This word or the adjective "REQUIRED" means that the item is an

requirement of the specification.

- SHOULD

  This word or the adjective "RECOMMENDED" means that there might exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before taking a different course.

- MAY

  This word or the adjective "OPTIONAL" means that this item is truly optional. One vendor might choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

5.0 SECURE HASH FUNCTION

The security of the OTP system is based on the non-invertability of a secure hash function. Such a function must be tractable to compute in the forward direction, but computationally infeasible to invert.

The interfaces are currently defined for three such hash algorithms, MD4 [2] and MD5 [6] by Ronald Rivest, and SHA [7] by NIST. All conforming implementations of both server and generators MUST support MD5. They SHOULD support SHA and MAY also support MD4. Clearly, the generator and server must use the same algorithm in order to interoperate. Other hash algorithms may be specified for use with this system by publishing the appropriate interfaces.

The secure hash algorithms listed above have the property that they accept an input that is arbitrarily long and produce a fixed size output. The OTP system folds this output to 64 bits using the algorithms in the Appendix A. 64 bits is also the length of the one-time passwords. This is believed to be long enough to be secure and short enough to be entered manually (see below, Form of Output) when necessary.

6.0 GENERATION OF ONE-TIME PASSWORDS

This section describes the generation of the one-time passwords. This process consists of an initial step in which all inputs are combined, a computation step where the secure hash function is applied a specified number of times, and an output function where the 64 bit one-time password is converted to a human readable form.

Appendix C contains examples of the outputs given a collection of inputs. It provides implementors with a means of verification the

of these algorithms.

Initial Step

In principle, the user's secret pass-phrase may be of any length. To reduce the risk from techniques such as exhaustive search or dictionary attacks, character string pass-phrases MUST contain at least 10 characters (see Form of Inputs below). All implementations MUST support a pass-phrases of at least 63 characters. The secret pass-phrase is frequently, but is not required to be, textual information provided by a user.

In this step, the pass phrase is concatenated with a seed that is transmitted from the server in clear text. This non-secret seed allows clients to use the same secret pass-phrase on multiple machines (using different seeds) and to safely recycle their secret pass-phrases by changing the seed.

The result of the concatenation is passed through the secure hash function and then is reduced to 64 bits using one of the function dependent algorithms shown in Appendix A.

Computation Step

A sequence of one-time passwords is produced by applying the secure hash function multiple times to the output of the initial step (called S). That is, the first one-time password to be used is produced by passing S through the secure hash function a number of times (N) specified by the user. The next one-time password to be used is generated by passing S though the secure hash function N-1 times. An eavesdropper who has monitored the transmission of a one-time password would not be able to generate the next required password because doing so would mean inverting the hash function.

Form of Inputs

The secret pass-phrase is seen only by the OTP generator. To allow interchangeability of generators, all generators MUST support a secret pass-phrase of 10 to 63 characters. Implementations MAY support a longer pass-phrase, but such implementations risk the loss of interchangeability with implementations supporting only the minimum.

The seed MUST consist of purely alphanumeric characters and MUST be of one to 16 characters in length. The seed is a string of characters that MUST not contain any blanks and SHOULD consist of strictly alphanumeric characters from the ISO-646 Invariant Code Set. The seed MUST be case insensitive and MUST be internally converted to lower case before it is processed.

The sequence number and seed together constitute a larger unit of

called the challenge. The challenge gives the generator the parameters it needs to calculate the correct one-time password from the secret pass-phrase. The challenge MUST be in a standard syntax so that automated generators can recognize the challenge in context and extract these parameters. The syntax of the challenge is:

otp-<algorithm identifier> <sequence integer> <seed>

The three tokens MUST be separated by a white space (defined as any number of spaces and/or tabs) and the entire challenge string MUST be terminated with either a space or a new line. The string "otp-" MUST be in lower case. The algorithm identifier is case sensitive (the existing identifiers are all lower case), and the seed is case insensitive and converted before use to lower case. If additional algorithms are defined, appropriate identifiers (short, but not limited to three or four characters) must be defined. The currently defined algorithm identifiers are:

md4	MD4 Message Digest
md5	MD5 Message Digest
sha1	NIST Secure Hash Algorithm Revision 1

An example of an OTP challenge is: otp-md5 487 dog2

Form of Output

The one-time password generated by the above procedure is 64 bits in length. Entering a 64 bit number is a difficult and error prone process. Some generators insert this password into the input stream and some others make it available for system "cut and paste." Still other arrangements require the one-time password to be entered manually. The OTP system is designed to facilitate this manual entry without impeding automatic methods. The one-time password therefore MAY be converted to, and all servers MUST be capable of accepting it as, a sequence of six short (1 to 4 letter) easily typed words that only use characters from ISO-646 IVCS. Each word is chosen from a dictionary of 2048 words; at 11 bits per word, all one-time passwords may be encoded.

The two extra bits in this encoding are used to store a checksum. The 64 bits of key are broken down into pairs of bits, then these pairs are summed together. The two least significant bits of this sum are encoded in the last two bits of the six word sequence with the least significant bit of the sum as the last bit encoded. All OTP generators MUST calculate this checksum and all OTP servers MUST verify this checksum explicitly as part of the operation of decoding this representation of the one-time password.

Generators that produce the six-word format MUST present the words

in upper case with single spaces used as separators. All servers MUST accept six-word format without regard to case and white space

as a separator. The two lines below represent the same one-time password. The first is valid as output from a generator and as input a server, the second is valid only as human input to a server.

```
OUST COAT FOAL MUG BEAK TOTE
oust coat foal  mug  beak  tote
```

Interoperability requires that all OTP servers and generators use the same dictionary. The standard dictionary was originally specified in the "S/KEY One Time Password System" that is described in RFC 1760 [5]. This dictionary is included in this document as Appendix D.

To facilitate the implementation of smaller generators, hexadecimal output is an acceptable alternative for the presentation of the one-time password. All implementations of the server software MUST accept case-insensitive hexadecimal as well as six-word format. The hexadecimal digits may be separated by white space so servers are REQUIRED to ignore all white space. If the representation is partitioned by white space, leading zeros must be retained. Examples of hexadecimal format are:

Representation	Value
3503785b369cda8b	0x3503785b369cda8b
e5cc a1b8 7c13 096b	0xe5cca1b87c13096b
C7 48 90 F4 27 7B A1 CF	0xc74890f4277ba1cf
47 9 A68 28 4C 9D 0 1BC	0x479a68284c9d01bc

In addition to accepting six-word and hexadecimal encodings of the 64 bit one-time password, servers SHOULD accept the alternate dictionary encoding described in Appendix B. The six words in this encoding MUST not overlap the set of words in the standard dictionary. To avoid ambiguity with the hexadecimal representation, words in the alternate dictionary MUST not be comprised solely of the letters A-F. Decoding words thus encoded does not require any knowledge of the alternative dictionary used so the acceptance of any alternate dictionary implies the acceptance of all alternate dictionaries. Words in the alternative dictionaries are case sensitive. Generators and servers MUST preserve the case in the processing of these words.

In summary, all conforming servers MUST accept six-word input that uses the Standard Dictionary (RFC 1760 and Appendix D), MUST accept hexadecimal encoding, and SHOULD accept six-word input that uses the Alternative Dictionary technique (Appendix B). As there is a remote possibility that a hexadecimal encoding of a one-time password will

look like a valid six-word standard dictionary encoding, all implementations MUST use the following scheme. If a six-word encoded one-time password is valid, it is accepted. Otherwise, if

one-time password can be interpreted as hexadecimal, and with that decoding it is valid, then it is accepted.

7.0 VERIFICATION OF ONE-TIME PASSWORDS

An application on the server system that requires OTP authentication is expected to issue an OTP challenge as described above. Given the parameters from this challenge and the secret pass-phrase, the generator can compute (or lookup) the one-time password that is passed to the server to be verified.

The server system has a database containing, for each user, the one-time password from the last successful authentication or the first OTP of a newly initialized sequence. To authenticate the user, the server decodes the one-time password received from the generator into a 64-bit key and then runs this key through the secure hash function once. If the result of this operation matches the stored previous OTP, the authentication is successful and the accepted one-time password is stored for future use.

8.0 PASS-PHRASE CHANGES

Because the number of hash function applications executed by the generator decreases by one each time, at some point the user must reinitialize the system or be unable to authenticate.

Although some installations may not permit users to initialize remotely, implementations MUST provide a means to do so that does not reveal the user's secret pass-phrase. One way is to provide a means to reinitialize the sequence through explicit specification of the first one-time password.

When the sequence of one-time passwords is reinitialized, implementations MUST verify that the seed or the pass-phrase is changed. Installations SHOULD discourage any operation that sends the secret pass-phrase over a network in clear-text as such practice defeats the concept of a one-time password.

Implementations MAY use the following technique for [re]initialization:

- The user picks a new seed and hash count (default values may be offered). The user provides these, along with the corresponding generated one-time password, to the host system.

- The user MAY also provide the corresponding generated one time password for count-1 as an error check.

- The user SHOULD provide the generated one-time password for

the old seed and old hash count to protect an idle terminal or workstation (this implies that when the count is 1, the

user can login but cannot then change the seed or count).

In the future a specific protocol may be defined for reinitialization that will permit smooth and possibly automated interoperation of all hosts and generators.

9.0 PROTECTION AGAINST RACE ATTACK

All conforming server implementations MUST protect against the race condition described in this section. A defense against this attack is outlined; implementations MAY use this approach or MAY select an alternative defense.

It is possible for an attacker to listen to most of a one-time password, guess the remainder, and then race the legitimate user to complete the authentication. Multiple guesses against the last word of the six-word format are likely to succeed.

One possible defense is to prevent a user from starting multiple simultaneous authentication sessions. This means that once the legitimate user has initiated authentication, an attacker would be blocked until the first authentication process has completed. In this approach, a timeout is necessary to thwart a denial of service attack.

10.0 SECURITY CONSIDERATIONS

This entire document discusses an authentication system that improves security by limiting the danger of eavesdropping/replay attacks that have been used against simple password systems [4].

The use of the OTP system only provides protections against passive eavesdropping/replay attacks. It does not provide for the privacy of transmitted data, and it does not provide protection against active attacks such as session hijacking that are known to be present in the current Internet [9]. The use of IP Security (IPsec), see [10], [11], and [12] is recommended to protect against TCP session hijacking.

The success of the OTP system to protect host systems is dependent on the non-invertability of the secure hash functions used. To our knowledge, none of the hash algorithms have been broken, but it is generally believed [6] that MD4 is not as strong as MD5. If a server supports multiple hash algorithms, it is only as secure as the weakest algorithm.

11.0 ACKNOWLEDGMENTS

The idea behind OTP authentication was first proposed by Leslie

Lamport [1]. Bellcore's S/KEY system, from which OTP is derived, was proposed by Phil Karn, who also wrote most of the Bellcore reference

12.0 REFERENCES

[1] Leslie Lamport, "Password Authentication with Insecure Communication", Communications of the ACM 24.11 (November 1981), 770-772

[2] R. L. Rivest, The MD4 Message-Digest Algorithm, "Request For Comments (RFC) 1320", MIT and RSA Data Security, Inc., April 1992

[3] Neil Haller, "The S/KEY One-Time Password System", Proceedings of the ISOC Symposium on Network and Distributed System Security, February 1994, San Diego, CA

[4] Neil Haller & Ran Atkinson, On Internet Authentication, "Request for Comments (RFC) 1704", Bellcore and Naval Research Laboratory, October 1994

[5] Neil Haller, The S/KEY One-Time Password System, "Request for Comments (RFC) 1760", Bellcore, February 1995

[6] R. L. Rivest, The MD5 Message-Digest Algorithm, "Request For Comments (RFC) 1321", MIT and RSA Data Security, Inc., April 1992

[7] National Institute of Standards and Technology (NIST), "Announcing the Secure Hash Standard", FIPS 180-1, U.S. Department of Commerce, April 1995.

[8] International Standard - Information Processing -- ISO 7-bit coded character set for information interchange (Invariant Code Set), ISO-646, International Standards Organization, Geneva, Switzerland, 1983

[9] Computer Emergency Response Team (CERT), "IP Spoofing and Hijacked Terminal Connections", CA-95:01, January 1995. Available via anonymous ftp from info.cert.org in /pub/cert_advisories.

[10] R. Atkinson, Security Architecture for the Internet Protocol, "Request for Comments (RFC) 1825", Naval Research Laboratory, August 1995

[11] R. Atkinson, IP Authentication Header, "Request for Comments (RFC) 1826", Naval Research Laboratory, August 1995

[12] R. Atkinson, IP Encapsulating Security Payload (ESP), "Request for Comments (RFC) 1827", Naval Research Laboratory, August

13.0 AUTHOR'S ADDRESS

Neil Haller
Bellcore
MCC 1C-265B
445 South Street
Morristown, NJ, 07960-6438, USA

Phone: +1 201 829-4478
Fax: +1 201 829-2504
Email: nmh@bellcore.com

Craig Metz
Kaman Sciences Corporation
For NRL Code 5544
4555 Overlook Avenue, S.W.
Washington, DC, 20375-5337, USA

Phone: +1 202 404-7122
Fax: +1 202 404-7942
Email: cmetz@cs.nrl.navy.mil

Philip J. Nesser II
Nesser & Nesser Consulting
13501 100th Ave NE
Suite 5202
Kirkland, WA 98034, USA

Phone: +1 206 481 4303
Email: pjnesser@martigny.ai.mit.edu

Mike Straw
Bellcore
RRC 1A-225
445 Hoes Lane
Piscataway, NJ 08854-4182

Phone: +1 908 699-5212
Email: mess@bellcore.com

Appendix A - Interfaces to Secure Hash Algorithms

Original interoperability tests provided valuable insights into the subtle problems which occur when converting protocol specifications into running code. In particular, the manipulation of bit ordered data is dependent on the architecture of the hardware, specifically the way in which a computer stores multi-byte data. The method is typically called big or little "endian." A big endian machine stores data with the most significant bit (msb) first, while a little endian machine stores the least significant bit (lsb) first. Thus, on a big endian machine data is stored left to right, while little endian machines store data right to left.

For example, the four byte value 0x11AABBCC is stored in a big endian machine as the following series of four bytes, "0x11", "0xAA", "0xBB", and "0xCC", while on a little endian machine the value would be stored as "0xCC", "0xBB", "0xAA", and "0x11".

For historical reasons, and to promote interoperability with existing implementations, it was decided that ALL hashes incorporated into the OTP protocol MUST store the output of their hash function in LITTLE ENDIAN format BEFORE the bit folding to 64 bits occurs. This is done in the implementations of MD4 and MD5 (see references [2] and [6]), while it must be explicitly done for the implementation of SHA1 (see reference [7]).

Any future hash functions implemented into the OTP protocol SHOULD provide a similar reference fragment of code to allow independent implementations to operate successfully.

MD4 Message Digest (see reference [2])

```
MD4_CTX md;
unsigned char result[16];

strcpy(buf, seed);     /* seed must be in lower case */
strcat(buf, passwd);
MD4Init(&md);
MD4Update(&md, (unsigned char *)buf, strlen(buf));
MD4Final(result, &md);

/* Fold the 128 bit result to 64 bits */
for (i = 0; i < 8; i++)
        result[i] ^= result[i+8];
```

MD5 Message Digest (see reference [6])

```
MD5_CTX md;
unsigned char result[16];
```

```
        strcpy(buf, seed);     /* seed must be in lower case */
        strcat(buf, passwd);
        MD5Init(&md);
        MD5Update(&md, (unsigned char *)buf, strlen(buf));
        MD5Final(result, &md);

        /* Fold the 128 bit result to 64 bits */
        for (i = 0; i < 8; i++)
                result[i] ^= result[i+8];
```

SHA Secure Hash Algorithm (see reference [7])

```
        SHA_INFO sha;
        unsigned char result[16];
        strcpy(buf, seed);     /* seed must be in lower case */
        strcat(buf, passwd);
        sha_init(&sha);
        sha_update(&sha, (unsigned char *)buf, strlen(buf));
        sha_final(&sha);       /* NOTE:  no result buffer */

        /* Fold the 160 bit result to 64 bits */
        sha.digest[0] ^= sha.digest[2];
        sha.digest[1] ^= sha.digest[3];
        sha.digest[0] ^= sha.digest[4];

        /*
         * copy the resulting 64 bits to the result buffer in little endian
         * fashion (analogous to the way MD4Final() and MD5Final() do).
         */
        for (i = 0, j = 0; j < 8; i++, j += 4)
        {
                result[j]   = (unsigned char)(sha.digest[i] & 0xff);
                result[j+1] = (unsigned char)((sha.digest[i] >> 8) & 0xff);
                result[j+2] = (unsigned char)((sha.digest[i] >> 16) & 0xff);
                result[j+3] = (unsigned char)((sha.digest[i] >> 24) & 0xff);
        }
```

Appendix B - Alternative Dictionary Algorithm

The purpose of alternative dictionary encoding of the OTP one-time password is to allow the use of language specific or friendly words. As case translation is not always well defined, the alternative dictionary encoding is case sensitive. Servers SHOULD accept this encoding in addition to the standard 6-word and hexadecimal encodings.

GENERATOR ENCODING USING AN ALTERNATE DICTIONARY

The standard 6-word encoding uses the placement of a word in the dictionary to represent an 11-bit number. The 64-bit one-time password can then be represented by six words.

An alternative dictionary of 2048 words may be created such that each word W and position of the word in the dictionary N obey the relationship:

alg(W) % 2048 == N

where

alg is the hash algorithm used (e.g. MD4, MD5, SHA1).

In addition, no words in the standard dictionary may be chosen.

The generator expands the 64-bit one-time password to 66 bits by computing parity as with the standard 6-word encoding. The six 11-bit numbers are then converted to words using the dictionary that was created such that the above relationship holds.

SERVER DECODING OF ALTERNATE DICTIONARY ONE-TIME PASSWORDS

The server accepting alternative dictionary encoding converts each word to an 11-bit number using the above encoding. These numbers are then used in the same way as the decoded standard dictionary words to form the 66-bit one-time password.

The server does not need to have access to the alternate dictionary that was used to create the one-time password it is authenticating. This is because the decoding from word to 11-bit number does not make any use of the dictionary. As a result of the independence of the dictionary, a server accepting one alternate dictionary accept all alternate dictionaries.

Appendix C - OTP Verification Examples

This appendix provides a series of inputs and correct outputs for all three of the defined OTP cryptographic hashes, specifically MD4, MD5, and SHA1. This document is intended to be used by developers for interoperability checks when creating generators or servers. Output is provided in both hexadecimal notation and the six word encoding documented in Appendix D.

GENERAL CHECKS

Note that the output given for these checks is not intended to be taken literally, but describes the type of action that should be taken.

Pass Phrase Length

```
 Input:
   Pass Phrase: Too_short
   Seed: iamvalid
   Count: 99
   Hash: ANY
 Output:
   ERROR:  Pass Phrase too short

 Input:
   Pass Phrase:
     1234567890123456789012345678901234567890123456789012345678901234
   Seed: iamvalid
   Count: 99
   Hash: ANY
 Output:
   WARNING: Pass Phrase longer than the recommended maximum length of
63
```

Seed Values

```
 Input:
   Pass Phrase:  A_Valid_Pass_Phrase
   Seed: Length_Okay
   Count: 99
   Hash: ANY
 Output:
   ERROR: Seed must be purely alphanumeric

 Input:
   Pass Phrase:  A_Valid_Pass_Phrase
   Seed: LengthOfSeventeen
```

```
   Count: 99
   Hash: ANY
```

```
 Output:
   ERROR: Seed must be between 1 and 16 characters in length

 Input:
   Pass Phrase:  A_Valid_Pass_Phrase
   Seed: A Seed
   Count: 99
   Hash: ANY
 Output:
   ERROR: Seed must not contain any spaces
```

Parity Calculations

```
 Input:
   Pass Phrase: A_Valid_Pass_Phrase
   Seed: AValidSeed
   Count: 99
   Hash: MD5
 Output:
   Hex: 85c43ee03857765b
   Six Word(CORRECT):          FOWL KID MASH DEAD DUAL OAF
   Six Word(INCORRECT PARITY): FOWL KID MASH DEAD DUAL NUT
   Six Word(INCORRECT PARITY): FOWL KID MASH DEAD DUAL O
   Six Word(INCORRECT PARITY): FOWL KID MASH DEAD DUAL OAK
```

MD4 ENCODINGS

Pass Phrase	Seed	Cnt	Hex	Six Word Format
This is a test.	TeSt	0	D185 4218 EBBB 0B51	ROME MUG FRED SCAN LIVE LACE
This is a test.	TeSt	1	6347 3EF0 1CD0 B444	CARD SAD MINI RYE COL KIN
This is a test.	TeSt	99	C5E6 1277 6E6C 237A	NOTE OUT IBIS SINK NAVE MODE
AbCdEfGhIjK	alpha1	0	5007 6F47 EB1A DE4E	AWAY SEN ROOK SALT LICE MAP
AbCdEfGhIjK	alpha1	1	65D2 0D19 49B5 F7AB	CHEW GRIM WU HANG BUCK SAID
AbCdEfGhIjK	alpha1	99	D150 C82C CE6F 62D1	ROIL FREE COG HUNK WAIT COCA
OTP's are good	correct	0	849C 79D4 F6F5 5388	FOOL STEM DONE TOOL BECK NILE
OTP's are good	correct	1	8C09 92FB 2508 47B1	GIST AMOS MOOT AIDS FOOD SEEM
OTP's are good	correct	99	3F3B F4B4 145F D74B	TAG SLOW NOV MIN WOOL KENO

MD5 ENCODINGS

Pass Phrase	Seed	Cnt	Hex	Six Word Format
This is a test.	TeSt	0	9E87 6134 D904 99DD	INCH SEA ANNE LONG AHEM TOUR
This is a test.	TeSt	1	7965 E054 36F5 029F	EASE OIL FUM CURE AWRY AVIS
This is a test.	TeSt	99	50FE 1962 C496 5880	BAIL TUFT BITS GANG CHEF THY
AbCdEfGhIjK	alpha1	0	8706 6DD9 644B F206	FULL PEW DOWN ONCE MORT ARC

```
AbCdEfGhIjK     alpha1   1  7CD3 4C10 40AD D14B FACT HOOF AT FIST SITE KENT
```

```
AbCdEfGhIjK     alpha1  99  5AA3 7A81 F212 146C BODE HOP JAKE STOW JUT RAP
OTP's are good  correct  0  F205 7539 43DE 4CF9 ULAN NEW ARMY FUSE SUIT EYED
OTP's are good  correct  1  DDCD AC95 6F23 4937 SKIM CULT LOB SLAM POE HOWL
OTP's are good  correct 99  B203 E28F A525 BE47 LONG IVY JULY AJAR BOND LEE
```

SHA1 ENCODINGS

```
Pass Phrase     Seed    Cnt Hex                 Six Word Format
================================================================================
This is a test. TeSt     0  BB9E 6AE1 979D 8FF4 MILT VARY MAST OK SEES WENT
This is a test. TeSt     1  63D9 3663 9734 385B CART OTTO HIVE ODE VAT NUT
This is a test. TeSt    99  87FE C776 8B73 CCF9 GAFF WAIT SKID GIG SKY EYED
AbCdEfGhIjK     alpha1   0  7B4C 5831 CCED CD36 LEST OR HEEL SCOT ROB SUIT
AbCdEfGhIjK     alpha1   1  D07C E229 B5CF 119B RITE TAKE GELD COST TUNE RECK
AbCdEfGhIjK     alpha1  99  27BC 7103 5AAF 3DC6 MAY STAR TIN LYON VEDA STAN
OTP's are good  correct  0  D51F 3E99 BF8E 6F0B RUST WELT KICK FELL TAIL FRAU
OTP's are good  correct  1  82AE B52D 9437 74E4 FLIT DOSE ALSO MEW DRUM DEFY
OTP's are good  correct 99  4F29 6A74 FE15 67EC AURA ALOE HURL WING BERG WAIT
```

Appendix D - Dictionary for Converting Between 6-Word and Binary Formats

This dictionary is from the module put.c in the original Bellcore reference distribution.

```
{       "A",     "ABE",   "ACE",   "ACT",   "AD",    "ADA",   "ADD",
"AGO",   "AID",   "AIM",   "AIR",   "ALL",   "ALP",   "AM",    "AMY",
"AN",    "ANA",   "AND",   "ANN",   "ANT",   "ANY",   "APE",   "APS",
"APT",   "ARC",   "ARE",   "ARK",   "ARM",   "ART",   "AS",    "ASH",
"ASK",   "AT",    "ATE",   "AUG",   "AUK",   "AVE",   "AWE",   "AWK",
"AWL",   "AWN",   "AX",   "AYE",   "BAD",   "BAG",   "BAH",   "BAM",
"BAN",   "BAR",   "BAT",   "BAY",   "BE",    "BED",   "BEE",   "BEG",
"BEN",   "BET",   "BEY",   "BIB",   "BID",   "BIG",   "BIN",   "BIT",
"BOB",   "BOG",   "BON",   "BOO",   "BOP",   "BOW",   "BOY",   "BUB",
"BUD",   "BUG",   "BUM",   "BUN",   "BUS",   "BUT",   "BUY",   "BY",
"BYE",   "CAB",   "CAL",   "CAM",   "CAN",   "CAP",   "CAR",   "CAT",
"CAW",   "COD",   "COG",   "COL",   "CON",   "COO",   "COP",   "COT",
"COW",   "COY",   "CRY",   "CUB",   "CUE",   "CUP",   "CUR",   "CUT",
"DAB",   "DAD",   "DAM",   "DAN",   "DAR",   "DAY",   "DEE",   "DEL",
"DEN",   "DES",   "DEW",   "DID",   "DIE",   "DIG",   "DIN",   "DIP",
"DO",    "DOE",   "DOG",   "DON",   "DOT",   "DOW",   "DRY",   "DUB",
"DUD",   "DUE",   "DUG",   "DUN",   "EAR",   "EAT",   "ED",    "EEL",
"EGG",   "EGO",   "ELI",   "ELK",   "ELM",   "ELY",   "EM",    "END",
"EST",   "ETC",   "EVA",   "EVE",   "EWE",   "EYE",   "FAD",   "FAN",
"FAR",   "FAT",   "FAY",   "FED",   "FEE",   "FEW",   "FIB",   "FIG",
"FIN",   "FIR",   "FIT",   "FLO",   "FLY",   "FOE",   "FOG",   "FOR",
"FRY",   "FUM",   "FUN",   "FUR",   "GAB",   "GAD",   "GAG",   "GAL",
"GAM",   "GAP",   "GAS",   "GAY",   "GEE",   "GEL",   "GEM",   "GET",
"GIG",   "GIL",   "GIN",   "GO",    "GOT",   "GUM",   "GUN",   "GUS",
"GUT",   "GUY",   "GYM",   "GYP",   "HA",    "HAD",   "HAL",   "HAM",
"HAN",   "HAP",   "HAS",   "HAT",   "HAW",   "HAY",   "HE",    "HEM",
"HEN",   "HER",   "HEW",   "HEY",   "HI",    "HID",   "HIM",   "HIP",
"HIS",   "HIT",   "HO",   "HOB",   "HOC",   "HOE",   "HOG",   "HOP",
"HOT",   "HOW",   "HUB",   "HUE",   "HUG",   "HUH",   "HUM",   "HUT",
"I",     "ICY",   "IDA",   "IF",    "IKE",   "ILL",   "INK",   "INN",
"IO",    "ION",   "IQ",   "IRA",   "IRE",   "IRK",   "IS",    "IT",
"ITS",   "IVY",   "JAB",   "JAG",   "JAM",   "JAN",   "JAR",   "JAW",
"JAY",   "JET",   "JIG",   "JIM",   "JO",    "JOB",   "JOE",   "JOG",
"JOT",   "JOY",   "JUG",   "JUT",   "KAY",   "KEG",   "KEN",   "KEY",
"KID",   "KIM",   "KIN",   "KIT",   "LA",    "LAB",   "LAC",   "LAD",
"LAG",   "LAM",   "LAP",   "LAW",   "LAY",   "LEA",   "LED",   "LEE",
"LEG",   "LEN",   "LEO",   "LET",   "LEW",   "LID",   "LIE",   "LIN",
"LIP",   "LIT",   "LO",   "LOB",   "LOG",   "LOP",   "LOS",   "LOT",
"LOU",   "LOW",   "LOY",   "LUG",   "LYE",   "MA",    "MAC",   "MAD",
"MAE",   "MAN",   "MAO",   "MAP",   "MAT",   "MAW",   "MAY",   "ME",
"MEG",   "MEL",   "MEN",   "MET",   "MEW",   "MID",   "MIN",   "MIT",
```

```
"MOB",   "MOD",   "MOE",   "MOO",   "MOP",   "MOS",   "MOT",   "MOW",
"MUD",   "MUG",   "MUM",   "MY",    "NAB",   "NAG",   "NAN",   "NAP",
"NAT",   "NAY",   "NE",   "NED",   "NEE",   "NET",   "NEW",   "NIB",
```

"NIL", "NIP", "NIT", "NO", "NOB", "NOD", "NON", "NOR",
"NOT", "NOV", "NOW", "NU", "NUN", "NUT", "O", "OAF",
"OAK", "OAR", "OAT", "ODD", "ODE", "OF", "OFF", "OFT",
"OH", "OIL", "OK", "OLD", "ON", "ONE", "OR", "ORB",
"ORE", "ORR", "OS", "OTT", "OUR", "OUT", "OVA", "OW",
"OWE", "OWL", "OWN", "OX", "PA", "PAD", "PAL", "PAM",
"PAN", "PAP", "PAR", "PAT", "PAW", "PAY", "PEA", "PEG",
"PEN", "PEP", "PER", "PET", "PEW", "PHI", "PI", "PIE",
"PIN", "PIT", "PLY", "PO", "POD", "POE", "POP", "POT",
"POW", "PRO", "PRY", "PUB", "PUG", "PUN", "PUP", "PUT",
"QUO", "RAG", "RAM", "RAN", "RAP", "RAT", "RAW", "RAY",
"REB", "RED", "REP", "RET", "RIB", "RID", "RIG", "RIM",
"RIO", "RIP", "ROB", "ROD", "ROE", "RON", "ROT", "ROW",
"ROY", "RUB", "RUE", "RUG", "RUM", "RUN", "RYE", "SAC",
"SAD", "SAG", "SAL", "SAM", "SAN", "SAP", "SAT", "SAW",
"SAY", "SEA", "SEC", "SEE", "SEN", "SET", "SEW", "SHE",
"SHY", "SIN", "SIP", "SIR", "SIS", "SIT", "SKI", "SKY",
"SLY", "SO", "SOB", "SOD", "SON", "SOP", "SOW", "SOY",
"SPA", "SPY", "SUB", "SUD", "SUE", "SUM", "SUN", "SUP",
"TAB", "TAD", "TAG", "TAN", "TAP", "TAR", "TEA", "TED",
"TEE", "TEN", "THE", "THY", "TIC", "TIE", "TIM", "TIN",
"TIP", "TO", "TOE", "TOG", "TOM", "TON", "TOO", "TOP",
"TOW", "TOY", "TRY", "TUB", "TUG", "TUM", "TUN", "TWO",
"UN", "UP", "US", "USE", "VAN", "VAT", "VET", "VIE",
"WAD", "WAG", "WAR", "WAS", "WAY", "WE", "WEB", "WED",
"WEE", "WET", "WHO", "WHY", "WIN", "WIT", "WOK", "WON",
"WOO", "WOW", "WRY", "WU", "YAM", "YAP", "YAW", "YE",
"YEA", "YES", "YET", "YOU", "ABED", "ABEL", "ABET", "ABLE",
"ABUT", "ACHE", "ACID", "ACME", "ACRE", "ACTA", "ACTS", "ADAM",
"ADDS", "ADEN", "AFAR", "AFRO", "AGEE", "AHEM", "AHOY", "AIDA",
"AIDE", "AIDS", "AIRY", "AJAR", "AKIN", "ALAN", "ALEC", "ALGA",
"ALIA", "ALLY", "ALMA", "ALOE", "ALSO", "ALTO", "ALUM", "ALVA",
"AMEN", "AMES", "AMID", "AMMO", "AMOK", "AMOS", "AMRA", "ANDY",
"ANEW", "ANNA", "ANNE", "ANTE", "ANTI", "AQUA", "ARAB", "ARCH",
"AREA", "ARGO", "ARID", "ARMY", "ARTS", "ARTY", "ASIA", "ASKS",
"ATOM", "AUNT", "AURA", "AUTO", "AVER", "AVID", "AVIS", "AVON",
"AVOW", "AWAY", "AWRY", "BABE", "BABY", "BACH", "BACK", "BADE",
"BAIL", "BAIT", "BAKE", "BALD", "BALE", "BALI", "BALK", "BALL",
"BALM", "BAND", "BANE", "BANG", "BANK", "BARB", "BARD", "BARE",
"BARK", "BARN", "BARR", "BASE", "BASH", "BASK", "BASS", "BATE",
"BATH", "BAWD", "BAWL", "BEAD", "BEAK", "BEAM", "BEAN", "BEAR",
"BEAT", "BEAU", "BECK", "BEEF", "BEEN", "BEER", "BEET", "BELA",
"BELL", "BELT", "BEND", "BENT", "BERG", "BERN", "BERT", "BESS",
"BEST", "BETA", "BETH", "BHOY", "BIAS", "BIDE", "BIEN", "BILE",
"BILK", "BILL", "BIND", "BING", "BIRD", "BITE", "BITS", "BLAB",
"BLAT", "BLED", "BLEW", "BLOB", "BLOC", "BLOT", "BLOW", "BLUE",
"BLUM", "BLUR", "BOAR", "BOAT", "BOCA", "BOCK", "BODE", "BODY",
"BOGY", "BOHR", "BOIL", "BOLD", "BOLO", "BOLT", "BOMB", "BONA",

```
"BOND",  "BONE",  "BONG",  "BONN",  "BONY",  "BOOK",  "BOOM",  "BOON",
"BOOT",  "BORE",  "BORG",  "BORN",  "BOSE",  "BOSS",  "BOTH",  "BOUT",
"BOWL",  "BOYD",  "BRAD",  "BRAE",  "BRAG",  "BRAN",  "BRAY",  "BRED",
```

"BREW", "BRIG", "BRIM", "BROW", "BUCK", "BUDD", "BUFF", "BULB",
"BULK", "BULL", "BUNK", "BUNT", "BUOY", "BURG", "BURL", "BURN",
"BURR", "BURT", "BURY", "BUSH", "BUSS", "BUST", "BUSY", "BYTE",
"CADY", "CAFE", "CAGE", "CAIN", "CAKE", "CALF", "CALL", "CALM",
"CAME", "CANE", "CANT", "CARD", "CARE", "CARL", "CARR", "CART",
"CASE", "CASH", "CASK", "CAST", "CAVE", "CEIL", "CELL", "CENT",
"CERN", "CHAD", "CHAR", "CHAT", "CHAW", "CHEF", "CHEN", "CHEW",
"CHIC", "CHIN", "CHOU", "CHOW", "CHUB", "CHUG", "CHUM", "CITE",
"CITY", "CLAD", "CLAM", "CLAN", "CLAW", "CLAY", "CLOD", "CLOG",
"CLOT", "CLUB", "CLUE", "COAL", "COAT", "COCA", "COCK", "COCO",
"CODA", "CODE", "CODY", "COED", "COIL", "COIN", "COKE", "COLA",
"COLD", "COLT", "COMA", "COMB", "COME", "COOK", "COOL", "COON",
"COOT", "CORD", "CORE", "CORK", "CORN", "COST", "COVE", "COWL",
"CRAB", "CRAG", "CRAM", "CRAY", "CREW", "CRIB", "CROW", "CRUD",
"CUBA", "CUBE", "CUFF", "CULL", "CULT", "CUNY", "CURB", "CURD",
"CURE", "CURL", "CURT", "CUTS", "DADE", "DALE", "DAME", "DANA",
"DANE", "DANG", "DANK", "DARE", "DARK", "DARN", "DART", "DASH",
"DATA", "DATE", "DAVE", "DAVY", "DAWN", "DAYS", "DEAD", "DEAF",
"DEAL", "DEAN", "DEAR", "DEBT", "DECK", "DEED", "DEEM", "DEER",
"DEFT", "DEFY", "DELL", "DENT", "DENY", "DESK", "DIAL", "DICE",
"DIED", "DIET", "DIME", "DINE", "DING", "DINT", "DIRE", "DIRT",
"DISC", "DISH", "DISK", "DIVE", "DOCK", "DOES", "DOLE", "DOLL",
"DOLT", "DOME", "DONE", "DOOM", "DOOR", "DORA", "DOSE", "DOTE",
"DOUG", "DOUR", "DOVE", "DOWN", "DRAB", "DRAG", "DRAM", "DRAW",
"DREW", "DRUB", "DRUG", "DRUM", "DUAL", "DUCK", "DUCT", "DUEL",
"DUET", "DUKE", "DULL", "DUMB", "DUNE", "DUNK", "DUSK", "DUST",
"DUTY", "EACH", "EARL", "EARN", "EASE", "EAST", "EASY", "EBEN",
"ECHO", "EDDY", "EDEN", "EDGE", "EDGY", "EDIT", "EDNA", "EGAN",
"ELAN", "ELBA", "ELLA", "ELSE", "EMIL", "EMIT", "EMMA", "ENDS",
"ERIC", "EROS", "EVEN", "EVER", "EVIL", "EYED", "FACE", "FACT",
"FADE", "FAIL", "FAIN", "FAIR", "FAKE", "FALL", "FAME", "FANG",
"FARM", "FAST", "FATE", "FAWN", "FEAR", "FEAT", "FEED", "FEEL",
"FEET", "FELL", "FELT", "FEND", "FERN", "FEST", "FEUD", "FIEF",
"FIGS", "FILE", "FILL", "FILM", "FIND", "FINE", "FINK", "FIRE",
"FIRM", "FISH", "FISK", "FIST", "FITS", "FIVE", "FLAG", "FLAK",
"FLAM", "FLAT", "FLAW", "FLEA", "FLED", "FLEW", "FLIT", "FLOC",
"FLOG", "FLOW", "FLUB", "FLUE", "FOAL", "FOAM", "FOGY", "FOIL",
"FOLD", "FOLK", "FOND", "FONT", "FOOD", "FOOL", "FOOT", "FORD",
"FORE", "FORK", "FORM", "FORT", "FOSS", "FOUL", "FOUR", "FOWL",
"FRAU", "FRAY", "FRED", "FREE", "FRET", "FREY", "FROG", "FROM",
"FUEL", "FULL", "FUME", "FUND", "FUNK", "FURY", "FUSE", "FUSS",
"GAFF", "GAGE", "GAIL", "GAIN", "GAIT", "GALA", "GALE", "GALL",
"GALT", "GAME", "GANG", "GARB", "GARY", "GASH", "GATE", "GAUL",
"GAUR", "GAVE", "GAWK", "GEAR", "GELD", "GENE", "GENT", "GERM",
"GETS", "GIBE", "GIFT", "GILD", "GILL", "GILT", "GINA", "GIRD",
"GIRL", "GIST", "GIVE", "GLAD", "GLEE", "GLEN", "GLIB", "GLOB",
"GLOM", "GLOW", "GLUE", "GLUM", "GLUT", "GOAD", "GOAL", "GOAT",
"GOER", "GOES", "GOLD", "GOLF", "GONE", "GONG", "GOOD", "GOOF",

```
"GORE",  "GORY",  "GOSH",  "GOUT",  "GOWN",  "GRAB",  "GRAD",  "GRAY",
"GREG",  "GREW",  "GREY",  "GRID",  "GRIM",  "GRIN",  "GRIT",  "GROW",
"GRUB",  "GULF",  "GULL",  "GUNK",  "GURU",  "GUSH",  "GUST",  "GWEN",
```

"GWYN", "HAAG", "HAAS", "HACK", "HAIL", "HAIR", "HALE", "HALF",
"HALL", "HALO", "HALT", "HAND", "HANG", "HANK", "HANS", "HARD",
"HARK", "HARM", "HART", "HASH", "HAST", "HATE", "HATH", "HAUL",
"HAVE", "HAWK", "HAYS", "HEAD", "HEAL", "HEAR", "HEAT", "HEBE",
"HECK", "HEED", "HEEL", "HEFT", "HELD", "HELL", "HELM", "HERB",
"HERD", "HERE", "HERO", "HERS", "HESS", "HEWN", "HICK", "HIDE",
"HIGH", "HIKE", "HILL", "HILT", "HIND", "HINT", "HIRE", "HISS",
"HIVE", "HOBO", "HOCK", "HOFF", "HOLD", "HOLE", "HOLM", "HOLT",
"HOME", "HONE", "HONK", "HOOD", "HOOF", "HOOK", "HOOT", "HORN",
"HOSE", "HOST", "HOUR", "HOVE", "HOWE", "HOWL", "HOYT", "HUCK",
"HUED", "HUFF", "HUGE", "HUGH", "HUGO", "HULK", "HULL", "HUNK",
"HUNT", "HURD", "HURL", "HURT", "HUSH", "HYDE", "HYMN", "IBIS",
"ICON", "IDEA", "IDLE", "IFFY", "INCA", "INCH", "INTO", "IONS",
"IOTA", "IOWA", "IRIS", "IRMA", "IRON", "ISLE", "ITCH", "ITEM",
"IVAN", "JACK", "JADE", "JAIL", "JAKE", "JANE", "JAVA", "JEAN",
"JEFF", "JERK", "JESS", "JEST", "JIBE", "JILL", "JILT", "JIVE",
"JOAN", "JOBS", "JOCK", "JOEL", "JOEY", "JOHN", "JOIN", "JOKE",
"JOLT", "JOVE", "JUDD", "JUDE", "JUDO", "JUDY", "JUJU", "JUKE",
"JULY", "JUNE", "JUNK", "JUNO", "JURY", "JUST", "JUTE", "KAHN",
"KALE", "KANE", "KANT", "KARL", "KATE", "KEEL", "KEEN", "KENO",
"KENT", "KERN", "KERR", "KEYS", "KICK", "KILL", "KIND", "KING",
"KIRK", "KISS", "KITE", "KLAN", "KNEE", "KNEW", "KNIT", "KNOB",
"KNOT", "KNOW", "KOCH", "KONG", "KUDO", "KURD", "KURT", "KYLE",
"LACE", "LACK", "LACY", "LADY", "LAID", "LAIN", "LAIR", "LAKE",
"LAMB", "LAME", "LAND", "LANE", "LANG", "LARD", "LARK", "LASS",
"LAST", "LATE", "LAUD", "LAVA", "LAWN", "LAWS", "LAYS", "LEAD",
"LEAF", "LEAK", "LEAN", "LEAR", "LEEK", "LEER", "LEFT", "LEND",
"LENS", "LENT", "LEON", "LESK", "LESS", "LEST", "LETS", "LIAR",
"LICE", "LICK", "LIED", "LIEN", "LIES", "LIEU", "LIFE", "LIFT",
"LIKE", "LILA", "LILT", "LILY", "LIMA", "LIMB", "LIME", "LIND",
"LINE", "LINK", "LINT", "LION", "LISA", "LIST", "LIVE", "LOAD",
"LOAF", "LOAM", "LOAN", "LOCK", "LOFT", "LOGE", "LOIS", "LOLA",
"LONE", "LONG", "LOOK", "LOON", "LOOT", "LORD", "LORE", "LOSE",
"LOSS", "LOST", "LOUD", "LOVE", "LOWE", "LUCK", "LUCY", "LUGE",
"LUKE", "LULU", "LUND", "LUNG", "LURA", "LURE", "LURK", "LUSH",
"LUST", "LYLE", "LYNN", "LYON", "LYRA", "MACE", "MADE", "MAGI",
"MAID", "MAIL", "MAIN", "MAKE", "MALE", "MALI", "MALL", "MALT",
"MANA", "MANN", "MANY", "MARC", "MARE", "MARK", "MARS", "MART",
"MARY", "MASH", "MASK", "MASS", "MAST", "MATE", "MATH", "MAUL",
"MAYO", "MEAD", "MEAL", "MEAN", "MEAT", "MEEK", "MEET", "MELD",
"MELT", "MEMO", "MEND", "MENU", "MERT", "MESH", "MESS", "MICE",
"MIKE", "MILD", "MILE", "MILK", "MILL", "MILT", "MIMI", "MIND",
"MINE", "MINI", "MINK", "MINT", "MIRE", "MISS", "MIST", "MITE",
"MITT", "MOAN", "MOAT", "MOCK", "MODE", "MOLD", "MOLE", "MOLL",
"MOLT", "MONA", "MONK", "MONT", "MOOD", "MOON", "MOOR", "MOOT",
"MORE", "MORN", "MORT", "MOSS", "MOST", "MOTH", "MOVE", "MUCH",
"MUCK", "MUDD", "MUFF", "MULE", "MULL", "MURK", "MUSH", "MUST",
"MUTE", "MUTT", "MYRA", "MYTH", "NAGY", "NAIL", "NAIR", "NAME",

```
"NARY",  "NASH",  "NAVE",  "NAVY",  "NEAL",  "NEAR",  "NEAT",  "NECK",
"NEED",  "NEIL",  "NELL",  "NEON",  "NERO",  "NESS",  "NEST",  "NEWS",
"NEWT",  "NIBS",  "NICE",  "NICK",  "NILE",  "NINA",  "NINE",  "NOAH",
```

"NODE", "NOEL", "NOLL", "NONE", "NOOK", "NOON", "NORM", "NOSE",
"NOTE", "NOUN", "NOVA", "NUDE", "NULL", "NUMB", "OATH", "OBEY",
"OBOE", "ODIN", "OHIO", "OILY", "OINT", "OKAY", "OLAF", "OLDY",
"OLGA", "OLIN", "OMAN", "OMEN", "OMIT", "ONCE", "ONES", "ONLY",
"ONTO", "ONUS", "ORAL", "ORGY", "OSLO", "OTIS", "OTTO", "OUCH",
"OUST", "OUTS", "OVAL", "OVEN", "OVER", "OWLY", "OWNS", "QUAD",
"QUIT", "QUOD", "RACE", "RACK", "RACY", "RAFT", "RAGE", "RAID",
"RAIL", "RAIN", "RAKE", "RANK", "RANT", "RARE", "RASH", "RATE",
"RAVE", "RAYS", "READ", "REAL", "REAM", "REAR", "RECK", "REED",
"REEF", "REEK", "REEL", "REID", "REIN", "RENA", "REND", "RENT",
"REST", "RICE", "RICH", "RICK", "RIDE", "RIFT", "RILL", "RIME",
"RING", "RINK", "RISE", "RISK", "RITE", "ROAD", "ROAM", "ROAR",
"ROBE", "ROCK", "RODE", "ROIL", "ROLL", "ROME", "ROOD", "ROOF",
"ROOK", "ROOM", "ROOT", "ROSA", "ROSE", "ROSS", "ROSY", "ROTH",
"ROUT", "ROVE", "ROWE", "ROWS", "RUBE", "RUBY", "RUDE", "RUDY",
"RUIN", "RULE", "RUNG", "RUNS", "RUNT", "RUSE", "RUSH", "RUSK",
"RUSS", "RUST", "RUTH", "SACK", "SAFE", "SAGE", "SAID", "SAIL",
"SALE", "SALK", "SALT", "SAME", "SAND", "SANE", "SANG", "SANK",
"SARA", "SAUL", "SAVE", "SAYS", "SCAN", "SCAR", "SCAT", "SCOT",
"SEAL", "SEAM", "SEAR", "SEAT", "SEED", "SEEK", "SEEM", "SEEN",
"SEES", "SELF", "SELL", "SEND", "SENT", "SETS", "SEWN", "SHAG",
"SHAM", "SHAW", "SHAY", "SHED", "SHIM", "SHIN", "SHOD", "SHOE",
"SHOT", "SHOW", "SHUN", "SHUT", "SICK", "SIDE", "SIFT", "SIGH",
"SIGN", "SILK", "SILL", "SILO", "SILT", "SINE", "SING", "SINK",
"SIRE", "SITE", "SITS", "SITU", "SKAT", "SKEW", "SKID", "SKIM",
"SKIN", "SKIT", "SLAB", "SLAM", "SLAT", "SLAY", "SLED", "SLEW",
"SLID", "SLIM", "SLIT", "SLOB", "SLOG", "SLOT", "SLOW", "SLUG",
"SLUM", "SLUR", "SMOG", "SMUG", "SNAG", "SNOB", "SNOW", "SNUB",
"SNUG", "SOAK", "SOAR", "SOCK", "SODA", "SOFA", "SOFT", "SOIL",
"SOLD", "SOME", "SONG", "SOON", "SOOT", "SORE", "SORT", "SOUL",
"SOUR", "SOWN", "STAB", "STAG", "STAN", "STAR", "STAY", "STEM",
"STEW", "STIR", "STOW", "STUB", "STUN", "SUCH", "SUDS", "SUIT",
"SULK", "SUMS", "SUNG", "SUNK", "SURE", "SURF", "SWAB", "SWAG",
"SWAM", "SWAN", "SWAT", "SWAY", "SWIM", "SWUM", "TACK", "TACT",
"TAIL", "TAKE", "TALE", "TALK", "TALL", "TANK", "TASK", "TATE",
"TAUT", "TEAL", "TEAM", "TEAR", "TECH", "TEEM", "TEEN", "TEET",
"TELL", "TEND", "TENT", "TERM", "TERN", "TESS", "TEST", "THAN",
"THAT", "THEE", "THEM", "THEN", "THEY", "THIN", "THIS", "THUD",
"THUG", "TICK", "TIDE", "TIDY", "TIED", "TIER", "TILE", "TILL",
"TILT", "TIME", "TINA", "TINE", "TINT", "TINY", "TIRE", "TOAD",
"TOGO", "TOIL", "TOLD", "TOLL", "TONE", "TONG", "TONY", "TOOK",
"TOOL", "TOOT", "TORE", "TORN", "TOTE", "TOUR", "TOUT", "TOWN",
"TRAG", "TRAM", "TRAY", "TREE", "TREK", "TRIG", "TRIM", "TRIO",
"TROD", "TROT", "TROY", "TRUE", "TUBA", "TUBE", "TUCK", "TUFT",
"TUNA", "TUNE", "TUNG", "TURF", "TURN", "TUSK", "TWIG", "TWIN",
"TWIT", "ULAN", "UNIT", "URGE", "USED", "USER", "USES", "UTAH",
"VAIL", "VAIN", "VALE", "VARY", "VASE", "VAST", "VEAL", "VEDA",
"VEIL", "VEIN", "VEND", "VENT", "VERB", "VERY", "VETO", "VICE",

```
"VIEW",  "VINE",  "VISE",  "VOID",  "VOLT",  "VOTE",  "WACK",  "WADE",
"WAGE",  "WAIL",  "WAIT",  "WAKE",  "WALE",  "WALK",  "WALL",  "WALT",
"WAND",  "WANE",  "WANG",  "WANT",  "WARD",  "WARM",  "WARN",  "WART",
```

```
"WASH",  "WAST",  "WATS",  "WATT",  "WAVE",  "WAVY",  "WAYS",  "WEAK",
"WEAL",  "WEAN",  "WEAR",  "WEED",  "WEEK",  "WEIR",  "WELD",  "WELL",
"WELT",  "WENT",  "WERE",  "WERT",  "WEST",  "WHAM",  "WHAT",  "WHEE",
"WHEN",  "WHET",  "WHOA",  "WHOM",  "WICK",  "WIFE",  "WILD",  "WILL",
"WIND",  "WINE",  "WING",  "WINK",  "WINO",  "WIRE",  "WISE",  "WISH",
"WITH",  "WOLF",  "WONT",  "WOOD",  "WOOL",  "WORD",  "WORE",  "WORK",
"WORM",  "WORN",  "WOVE",  "WRIT",  "WYNN",  "YALE",  "YANG",  "YANK",
"YARD",  "YARN",  "YAWL",  "YAWN",  "YEAH",  "YEAR",  "YELL",  "YOGA",
"YOKE"   };
```