

## A Convention for Human-Readable 128-bit Keys

### Status of this Memo

Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. Note that other groups may also distribute working documents as Internet Drafts.

Internet Drafts are draft documents valid for a maximum of six months. This Internet Draft expires on February 20, 1995. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress."

Please check the I-D abstract listing contained in each Internet Draft directory to learn the current status of this or any other Internet Draft.

Distribution of this memo is unlimited.

### Introduction

The Internet community has begun to address matters of security. Recent standards, including version 2 of SNMP [[GM93](#)], and version 6 of IP [[Atk94](#)] have explicit requirements for an authentication mechanism. Both require use of a keyed message-digest algorithm, MD5 [[Riv92](#)]. Both require a key size of 128-bits. A 128-bit key, while sufficiently strong, is hard for most people to read, remember, and type in.

### A Solution Already Exists

The S/Key(tm) one-time password system [[Hal94](#)] uses MD4 (and now MD5, as well) to compute one-time passwords. It takes the 128-bit result of MD4 and collapses it to a 64-bit result. Despite the size reduction, 64-bit one-time passwords are still difficult for ordinary people to remember and enter. The authors of S/Key devised a system to make the 64-bit one-time password easy for people to enter.

Their idea was to transform the password into a string of small English words. English words are significantly easier for people to

INTERNET-DRAFT

Human-Readable 128-bit Keys

20 August 1994

both remember and type. The authors of S/Key started with a dictionary of 2048 English words, ranging in length from one to four characters. The space covered by a 64-bit key ( $2^{64}$ ) could be covered by six words from this dictionary ( $2^{66}$ ) with room remaining for parity. For example, an S/Key one-time password of hex value:

EB33 F77E E73D 4053

would become the following six English words:

TIDE ITCH SLOW REIN RULE MOT

### The Proposal

The code (see [Appendix A](#)) which S/Key uses to convert 64-bit numbers to six English words contains two primitives which perform conversions either way. The primitive `btoe(char *engout, char *c)` takes a 64-bit quantity referenced by `c` and places English words in the string referenced by `engout`. The primitive `etob(char *out, char *e)` performs the opposite with an input string of English words referenced by `e`, and by placing the 64-bit result into the buffer referenced by `out`.

The aforementioned primitives can be applied to both halves of a 128-bit key, or both halves of a string of twelve English words. Two new primitives (see [Appendix B](#)), `key2eng(char *engout, char *key)` and `eng2key(char *keyout, char *eng)` serve as wrappers which call the S/Key primitives twice, once for each half of the 128-bit key or string of twelve words.

For example, the 128-bit key of:

CCAC 2AED 5910 56BE 4F90 FD44 1C53 4766

would become

RASH BUSH MILK LOOK BAD BRIM AVID GAFF BAIT ROT POD LOVE

Likewise, a user should be able to type in

TROD MUTE TAIL WARM CHAR KONG HAAG CITY BORE O TEAL AWL

as a key, and the machine should make the translation to:

If this proposal is to work, it is critical that the dictionary of English words does not change with different implementations. A

McDonald

[Page 2]

---

INTERNET-DRAFT

Human-Readable 128-bit Keys

20 August 1994

freely redistributable reference implementation is given in Appendices A and B.

### Security Considerations

This document recommends a method of representing 128-bit keys using strings of English words. Since the strings of English words are easy to remember, people may potentially construct easy-to-guess strings of English words. With easy-to-guess strings comes the possibility of a sentential equivalent of a dictionary attack. In order to maximize the strength of any authentication mechanism that uses 128-bit keys, the keys must be sufficiently obscure. In particular, people should avoid the temptation to devise sentences.

### References

[Atk94] Atkinson, Randall, "IPv6 Authentication Header", Internet-Draft, August 1994.

[GM93] Galvin, J. and McCloghrie, K., "Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2)", [RFC1446](#), Internet Architecture Board, April 1993.

[Hal94] Haller, Neil M., "The S/Key(tm) One-Time Password System", Proceedings of the Symposium on Network & Distributed Systems Security, Internet Society, San Diego, February 1994.

[Riv92] Rivest, R., "The MD5 Message-Digest Algorithm", [RFC1321](#), Internet Architecture Board, April 1992.

### Author's Address

Daniel L. McDonald  
United States Naval Research Laboratory  
Code 5544  
4555 Overlook Ave. SW

Washington, DC 20375

Phone: (202) 404-7122

E-mail: danmcd@itd.nrl.navy.mil

McDonald

[Page 3]

---

INTERNET-DRAFT

Human-Readable 128-bit Keys

20 August 1994

#### Appendix A - Source for S/Key 8-bytes to/from Words Routines (put.c)

```
/* This code originally appeared in the source for S/Key, available in the
 * directory
 * ftp://thumper.bellcore.com/pub/nmh
 *
 * It has been modified only to remove explicit S/Key references.
 */
```

```
#include <stdio.h>
#include <string.h>
#include <assert.h>
#include <ctype.h>
```

```
#ifdef __STDC__
#define __ARGS(x) x
#else
#define __ARGS(x) ()
#endif
```

```
static unsigned long extract __ARGS((char *s,int start,int length));
static void standard __ARGS((char *word));
static void insert __ARGS((char *s, int x, int start, int length));
static int wsrch __ARGS((char *w,int low,int high));
```

```
/* Dictionary for integer-word translations */
char Wp[2048][4] = {
"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",

McDonald

[Page 4]

INTERNET-DRAFT

Human-Readable 128-bit Keys

20 August 1994

"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", "AHAM", "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", "HOB0", "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", "WISE", "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"  
};
```

```
/* Encode 8 bytes in 'c' as a string of English words.
```

```
 * Returns a pointer to a static buffer
```

```
 */
```

```
char *
```

```
btoe(engout,c)
```

```
char *c, *engout;
```

```
{
```

```
    char cp[9];      /* add in room for the parity 2 bits*/  
    int p,i ;
```

```
    engout[0] = '\0';
```

```
    memcpy(cp, c,8);
```

```
    /* compute parity */
```

```
    for(p = 0,i = 0; i < 64;i += 2)  
        p += extract(cp,i,2);
```

```
    cp[8] = (char)p << 6;
```

```
    strncat(engout,&Wp[extract(cp, 0,11)][0],4);
```

```
    strcat(engout," ");
```

```
    strncat(engout,&Wp[extract(cp,11,11)][0],4);
```

```
    strcat(engout," ");
```

```
    strncat(engout,&Wp[extract(cp,22,11)][0],4);
```

```
    strcat(engout," ");
```

```
    strncat(engout,&Wp[extract(cp,33,11)][0],4);
```

```
    strcat(engout," ");
```

```
    strncat(engout,&Wp[extract(cp,44,11)][0],4);
```

```
    strcat(engout," ");
```

```
    strncat(engout,&Wp[extract(cp,55,11)][0],4);
```

```
#ifdef notdef
```

```
    printf("engout is %s\n\r",engout);
```

```
#endif
```

```
    return(engout);
```

```
}
```

```
/* convert English to binary
```

```
 * returns 1 OK - all good words and parity is OK
```

```
 *      0 word not in data base
```

```
 *     -1 badly formed in put ie > 4 char word
```

```
 *     -2 words OK but parity is wrong
```

```
 */
```

```
int
```

```
etob(out, e)
char *out;
```

```
char *e;
{
    char *word;
    int i, p, v, l, low, high;
    char b[9];
    char input[36];

    if(e == NULL)
        return -1;

    strncpy(input, e, sizeof(input));
    memset(b, 0, sizeof(b));
    memset(out, 0, 8);
    for(i=0, p=0; i<6; i++, p+=11){
        if((word = strtok(i == 0 ? input : NULL, " ")) == NULL)
            return -1;
        l = strlen(word);
        if(l > 4 || l < 1){
            return -1;
        } else if(l < 4){
            low = 0;
            high = 570;
        } else {
            low = 571;
            high = 2047;
        }
        standard(word);
        if( (v = wsrch(word, low, high)) < 0 )
            return 0;
        insert(b, v, p, 11);
    }

    /* now check the parity of what we got */
    for(p = 0, i = 0; i < 64; i +=2)
        p += extract(b, i, 2);

    if( (p & 3) != extract(b, 64, 2) )
        return -2;
```

```

        memcpy(out,b,8);

        return 1;
}
/* Display 8 bytes as a series of 16-bit hex digits */
char *
put8(out,s)
char *out;
char *s;

```

McDonald

[Page 10]

INTERNET-DRAFT

Human-Readable 128-bit Keys

20 August 1994

```

{
    sprintf(out,"%02X%02X %02X%02X %02X%02X %02X%02X",
            s[0] & 0xff,s[1] & 0xff,s[2] & 0xff,
            s[3] & 0xff,s[4] & 0xff,s[5] & 0xff,
            s[6] & 0xff,s[7] & 0xff);
    return out;
}
#ifdef notdef
/* Encode 8 bytes in 'cp' as stream of ascii letters.
 * Provided as a possible alternative to btoc()
 */
char *
btoc(cp)
char *cp;
{
    int i;
    static char out[31];

    /* code out put by characters 6 bits each added to 0x21 (!)*/
    for(i=0;i <= 10;i++){
        /* last one is only 4 bits not 6*/
        out[i] = '!' + extract(cp,6*i,i >= 10 ? 4:6);
    }
    out[i] = '\0';
    return(out);
}
#endif

/* Internal subroutines for word encoding/decoding */

/* Dictionary binary search */
static int

```

```

wsrch(w,low,high)
char *w;
int low, high;
{
    int i,j;

    for(;;){
        i = (low + high)/2;
        if((j = strncmp(w,Wp[i],4)) == 0)
            return i;          /* Found it */
        if(high == low+1){
            /* Avoid effects of integer truncation in /2 */
            if(strncmp(w,Wp[high],4) == 0)
                return high;
            else
                return -1;
        }
    }
}

```

McDonald

[Page 11]

INTERNET-DRAFT

Human-Readable 128-bit Keys

20 August 1994

```

    }
    if(low >= high)
        return -1;          /* I don't *think* this can happen...*/
    if(j < 0)
        high = i;           /* Search lower half */
    else
        low = i;            /* Search upper half */
}

static void
insert(s, x, start, length)
char *s;
int x;
int start, length;
{
    unsigned char cl;
    unsigned char cc;
    unsigned char cr;
    unsigned long y;
    int shift;

    assert(length <= 11);
    assert(start >= 0);
    assert(length >= 0);
    assert(start + length <= 66);
}

```

```

shift = ((8 - ((start + length) % 8)) % 8);
y = (long) x << shift;
cl = (y >> 16) & 0xff;
cc = (y >> 8) & 0xff;
cr = y & 0xff;
if(shift + length > 16){
    s[start/8] |= cl;
    s[start/8 + 1] |= cc;
    s[start/8 + 2] |= cr;
} else if(shift + length > 8){
    s[start/8] |= cc;
    s[start/8 + 1] |= cr;
} else {
    s[start/8] |= cr;
}
}

```

```

static void
standard(word)
register char *word;
{
    while(*word){

```

```

        if(!isascii(*word))
            break;
        if(islower(*word))
            *word = toupper(*word);
        if(*word == '1')
            *word = 'L';
        if(*word == '0')
            *word = 'O';
        if(*word == '5')
            *word = 'S';
        word++;
    }
}

```

```

/* Extract 'length' bits from the char array 's' starting with bit 'start' */
static unsigned long
extract(s, start, length)
char *s;

```

```

int start, length;
{
    unsigned char cl;
    unsigned char cc;
    unsigned char cr;
    unsigned long x;

    assert(length <= 11);
    assert(start >= 0);
    assert(length >= 0);
    assert(start +length <= 66);

    cl = s[start/8];
    cc = s[start/8 +1];
    cr = s[start/8 +2];
    x = ((long)(cl<<8 | cc) <<8 | cr) ;
    x = x >> (24 - (length + (start %8)));
    x =( x & (0xffff >> (16-length) )    );
    return(x);
}

```

## Appendix B - Source for 128-bit key to/from English words (convert.c)

```

/* convert.c  --  Wrapper to S/Key binary-to-English routines.
                  Daniel L. McDonald  --  U. S. Naval Research Laboratory. */

#include <string.h>

/* eng2key() assumes words must be separated by spaces only.

   eng2key() returns

```

```

    1 if succeeded
    0 if word not in dictionary
    -1 if badly formed string
    -2 if words are okay but parity is wrong.
    (see etob() in S/Key)
*/

int eng2key(keyout,eng)
char *keyout,*eng;
{
    int rc=0,state=1;
    char *eng2;

    /* Find pointer to word 7. */

    for (eng2 = eng; rc<7 && (*(++eng2) != '\0'); )
        if (*eng2 != ' ')
        {
            rc += state;
            state = 0;
        }
        else state=1;

    if ( (rc = etob(keyout,eng)) != 1)
        return rc;

    rc = etob(keyout+8,eng2);

    return rc;
}

/* key2eng() assumes string referenced by engout has at least 60 characters
   (4*12 + 11 spaces + '\0') of space.

   key2eng() returns pointer to engout.
*/

```

```

char *key2eng(engout,key)
char *engout,*key;
{
    btoe(engout,key);
}

```



```
    strcat(engout," ");  
    btoe(engout+strlen(engout),key+8);  
}
```