Alessandro Levi Montalcini
C.so Re Umberto 10
10121 Torino
Italy
e-mail: LMontalcini@pmn.it
anonymous ftp: //ftp.alpcom.it/software/mac/LMontalcini

lex's Encrypt 1.0 documentation - April 1995

• Shareware: $5

◊ This utility is distributed as shareware: if you like it please honor the shareware system by sending $5 to the author at the address above. Since cashing international cheques is very expensive, please send cash only (U.S. dollars are best, but any other currency is fine).

◊ If you work on the Macintosh, there are many things you can send me instead of the money. Here are some suggestions: original software (but don't send any shareware/freeware unless you're the author), books or computer-related magazines (any free issue or subscription is welcome), CD-ROMs, any piece of new or used hardware, or anything else you've created.

◊ There's still another thing you could do: I'll finish my physics studies sometime in 1996 and I'm looking for a one-year job as a programmer (or maybe a scientist?) outside of Italy. Let me know if you have anything to offer.

◊ If you can't afford any of the above an e-mail, a postcard or a letter is welcome. I speak English, French and Italian. Please forgive me if I don't answer to all of your letters - I am often overloaded with my school work.

• Introduction

◊ Alex's Encrypt is a simple encryption utility that encrypts one file at a time. The encryption key is stored in a variable-length key file that can be as small as 512 bytes or as large as the entire file to encrypt.

◊ Using a key file instead of a password makes the encryption only suitable for cases where the key can be stored in a different location or sent separately to the receiver. I built it so that I could store large encrypted files in my ftp directory and send relatively small key files by e-mail to the people that had to download them. Password protection on the compressed archives was not enough, since it still allowed everyone to peek at the archive's contents.

◊ Obviously, keeping the key file and the encrypted file in the same place (or sending them together) makes the encryption completely useless. Which means that this program can't be used to encrypt data on your hard disk, unless you keep the key files on a separate floppy. If you do so, however, keep in mind that losing the floppy or the data on it means losing the encrypted files as well (there is no way to recover the encrypted files without their key file).

◊ You should give meaningful names to your key files, since only the original key file used for encryption can later be used to decrypt a file. If you have many key files with weird names, you won't be able to find the right one for a particular encrypted file.

◊ One last note: Alex's Encrypt reads the whole file in RAM before processing it, so there must be enough free space either in its own memory partition or in the system's unused memory for a file to be encrypted or decrypted.

• Safety considerations

◊ The encryption method used by Alex's Encrypt is pretty simple, but it can be unbreakable if it's used properly. The main thing to remember is that you should never encrypt files with long sequences of repeated data (like a very long series of spaces or zeroes) with a short key file. The minimum length of the key file is a complex argument which depends on the situation, but in all cases the key file should be longer than the longest sequence of repeated data in the file to encrypt.

◊ If the key file is as long as the file to encrypt, then the encryption scheme can't be broken in any way (regardless of the contents of the file to encrypt). However, you end up with two large files instead of one, which is not often useful. This kind of protection can be used to split some very confidential information in two files, since each file doesn't carry any information unless they both come together; you could give each file to a different person so that they can't recover the original data until they meet.

◊ Using smaller key files (from 512 bytes to a few K) is almost as safe if you're encrypting files with little redundance (i.e. files with no repeated sequences). A compressed archive, for example, has almost no redundance; you can encrypt a StuffIt or Compact Pro archive with a short key file with very little impact on the safety level of the encryption scheme.

• Technical information

◊ An encryption scheme's safety level should always be measured assuming that the cracker knows everything except the key, so here's some information about the internal workings this program; you don't have to read it if you're not interested in the details of encryption.

◊ Alex's Encrypt uses the exclusive-or (xor) function to encrypt and decrypt the data; the key file is xor'ed to the source file repeatedly, until the end of the file is reached. Both the data and resource forks of the source file are encrypted and stored in the data fork of the encrypted file.

◊ The encrypted file has a small header that carries the original file's type and creator and some Finder information, so that they can be restored when the file is decrypted. There's also a checksum value that's used to make sure that the right key file is used for decryption; if the checksum doesn't match when the file is decrypted, then the wrong key file was used and the program refuses to restore the decrypted file.

◊ A key file is generated by xor'ing pseudo-random values (returned by the Random() toolbox call) to an arbitrary input file. The arbitrary input file provides the actual randomness of the key file, whereas the xor'ed Random values (which are not random at all since they are generated by an algorythm) are used to reduce the arbitrary file's redundance to near zero and to make the key file unreadable. This makes it possible to use just about any file as the arbitrary input file, since its data can't be restored. Besides, not all of the input file is used to generate the key file; Alex's Encrypt only reads a number of bytes equal to the desired key size from the data or resource fork, whichever is larger.

◊ Just to make things a bit more complicated, the seed value used for the Random() calls is computed from various time-dependant and machine-specific variables; the key file's creation date is always set to Jan 1, 1904 so that the time-dependant variables can't be guessed by looking at it. This makes it very hard to recover the arbitrary input file from the key file, although it has no effect on the safety of the final encrypted archives.

◊ A note on encrypting compressed archives (and other formatted files) with small key files: the only weak point here is the archive header, since part of the original header's contents (say, the file type and creator) could be guessed by someone who knows the format of the archive. This could make him recover a few bytes of the key file, but these are completely useless if he doesn't know the size of the key file; and even assuming he knows it, he could only recover a few bytes of the original archive for every chunk of encrypted data, where each chunk has the size of the key file. But then, a compressed archive carries encoded data that can't be recovered unless the archive is mostly or completely intact. In other words, there's still no way to break the encryption.

• Warning

◊ Don't call me if you lose a key file. There's no way to recover your data…

• Version history

◊ 1.0 - First public release.

• Distribution

◊ Alex's Encrypt is ©1994-95 Alessandro Levi Montalcini. It can be freely distributed as long as it is not modified and there's no charge for it, but it may not be included in any commercial package without my consent.

◊ You may find the latest version of all my shareware programs by anonymous ftp to ftp.alpcom.it, inside the /software/mac/LMontalcini directory. The complete ShareDisk package, which contains all my stuff and can be registered at a very low price, is also available there.

◊ All online services and bulletin boards may make it available to their users at no charge other than the normal connection fees.

◊ All non-profit user groups may distribute it at no charge.

◊ All magazines may publish it on floppy disk without asking me first, as long as I get a copy of the issue containing my software.

◊ All CD-ROM shareware collections and CD-ROM magazines may include it without my prior consent, as long as I get either a copy of the CD-ROM or an offer to buy the CD-ROM at a discounted price.

◊ All redistribution companies such as Educorp may distribute it, as long as I get a copy of each media containing my software and a catalog of the company's offerings (where applicable).

• Disclaimer

◊ Alex's Encrypt shouldn't cause any damage, but you're using it at your own risk. As an independent software developer, I can make no warranties whatsoever on it.

• Have fun!

◊ And don't forget to $$$ send your contribution $$$ so that more cool utilities will see the light in the near future, at the low-low-low costs of shareware.