About SandTiger

SandTiger is a security application that encrypts multiple files into a single archive. It uses powerful encryption algorithms like CAST-128, Blowfish and Diamond2, along with renouned hashes like SHA-1, MD5 and RIPEM-160. With SandTiger, you have the ability to archive an unlimited number of files, while maintaining their relative directories. You also have the option of enhancing your files by compressing them before encrypting to reduce redundant code. SandTiger supports multiple passwords in a single archive and has the ability to scramble filenames, hiding them from hex viewers.

Version 2.0 comes equipped with 3 encryption algorithms

- Blowfish, a symmetric block cipher that can be used as a drop-in replacement for DES or IDEA. It takes a variable-length key, from 32 bits to 448 bits, making it ideal for both domestic and exportable use.

- CAST-128 algorithm supports variable key lengths, anywhere from 40 bits to 128 bits in length. This ensures that an appropriate security level is given to data for the intended purpose and enables seamless interoperation with exportable versions of products, where necessary. CAST uses a 64-bit block size which is the same as the Data Encryption Standard (DES), making it a suitable drop-in replacement. CAST has been shown to be two to three times faster than a typical implementation of DES and six to nine times faster than a typical implementation of triple-DES.

The Diamond2 Block Cipher is a royalty-free, symmetric-key encryption algorithm based on a combination of nonlinear functions. Diamond uses a block size of 128 bits and a variable length key.

Credits

- Diamond2 implementation - by Micheal Paul Johnson. (DLOCK2)
- Blowfish implementation - by Eric Young. (SSLeay)
- Random Number Generator - by Matsumoto and Nishimura. (GENRAND)
- Implementation of CAST-128 is copyright 1996 Peter Gutmann, minor modifications for use with FastCAST by Leonard Janke
- Another CAST-128 library by Andrew E. Mileski <aem@netcom.ca> was used during the beta process

Features

- Blowfish encryption - up to 448bits key length, 64bit blocksize - *SSLeay 0.8.0 library*
- Featuring the strong CAST-128 encryption - up to 128bits key length, 64bit blocksize - *fastcast library*
- Powerful Diamond2 encryption - variable key length, 128bit blocksize - *Dlock2*
- Your choice of MD5, SHA-1 and RIPEM hashes
- Zlib Compression that compresses better than pkzip.
- Encrypt files and Entire directories
- Support for multiple passwords in a single archive
- Scramble filenames to hide them from hex viewers
- Designed for Microsoft Windows95 and WinNT 4.0
- Install/Uninstall Support
- OLE drag and drop

License/Copyright/Warranty

License/Copyright/Warranty

SandTiger(R) version 2.0

Contact Information

e-mail: bishop@ottawa.com
web site: homepage: http://members.tripod.com/~subrosa
snail mail: Selom Ofori
144A Woodridge Crescent
Nepean, Ontario
K2B 7S9
Canada

For ordering information, see the file ORDER.TXT.

License Agreement

You should carefully read the following terms and conditions before using this software. Your use of this software indicates your acceptance of this license agreement and warranty.

This program is copyrighted by the author. You may USE and distribute SandTiger as long as it is in accordance with CANADIAN LAW and the distribution archive remains intact, without any changes or modifications. The distribution archive is in ZIP format; however, you may convert the archive to any format you choose, so long as the above requirements are met.

Registered Version

One registered copy of SandTiger may either be used by a single person who uses the software personally on one or more computers, or installed on a single workstation used non-simultaneously by multiple people, but not both.

You may access the registered version of SandTiger through a network, provided that you have obtained individual licenses for the software covering all workstations that will access the software through the network. For instance, if 8 different workstations will access SandTiger on the network, each workstation must have its own SandTiger license, regardless of whether they use SandTiger at different times or concurrently.

Governing Law

This agreement shall be governed by the laws of the Province of Ontario

Disclaimer of Warranty

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Good data processing procedure dictates that any program be thoroughly tested with non-critical data before relying on it. The user must assume the entire risk of using the program. ANY LIABILITY OF THE SELLER WILL BE LIMITED EXCLUSIVELY TO PRODUCT REPLACEMENT OR REFUND OF PURCHASE PRICE.

Distribution

SandTiger may be distributed electronically only, and should reside on servers located and operated in the United States and/or Canada only.

Ordering the full version

The full version is available only to customers in the United States
and Canada. International orders will be rejected.

Registration entitles you to e-mail support and upgrades for a year.
Single user copies are $30.00 each. All price quotes in Canadian Currency.

SandTiger 2.0 Single Copy ____ copies at $30 each

Total payment = _____

NAME :
_____

COMPANY : _____

STREET :
_____

STREET :
_____

CITY :
_____

STATE/PROVINCE : _____ POSTAL CODE : _____

COUNTRY : _____

TELEPHONE : _____

E-mail Address : _____ (required) please verify by mailing to

bishop@ottawa.com

Cheques must be made payable to Selom Ofori in Canadian dollars drawn on a
Canadian bank.

Make Cheques/Money order payable to: Selom Ofori.
Mail to: Selom Ofori
144A Woodridge Crescent
Nepean, Ontario
K2B 7S9

A valid serial number will be sent via e-mail and snail mail after a successful purchase.

Contact Information
e-mail: bishop@ottawa.com
web site: homepage: http://members.tripod.com/~subrosa
snail mail: Selom Ofori
144A Woodridge Crescent
Nepean, Ontario
K2B 7S9
Canada

WHAT'S NEW - SandTiger version 2.0

- Most of the time was spent on including 'forgiveness'. Eg. if the disk
  becomes full, it asks to retry or abort and won't destroy or ruin the
  archive. Memory leaks, debug info and stuff like that has been taken out

- This is what I intended to create when I started working on this project
  It has been polished and is now out of beta. This is the real thing

- Newer files will always replace older files in an archive.

WHAT'S NEW - SandTiger version 1.6 FULL RELEASE

- This is a full release. CAST-128, Blowfish and Diamond2. You need a serial
  key to install sandtiger.
- Option included to scramble filenames.
- fixed a bug with RIPEMD that caused Sandtiger to refuse to decrypt your files

WHAT'S NEW - SandTiger version 1.6

- CAST-128 has been implemented and is now the default cipher
- Sandtiger now sets the file time to the original after extraction
- fixed the multiple redraw "feature" in v1.5
- fixed the 1000+ icon resource hog
- SandTiger is now released as a demo. 6 character password limit with 3 ciphers
  CAST-128(default), Blowfish(disabled in demo) and Diamond2(disabled in demo).
- compatible with version 1.5

WHAT'S NEW - SandTiger version 1.5

- Zlib compression with Z_BEST_COMPRESSION flag only. It compresses better than
  pkzip, but it's much slower.
- Diamond2 and Blowfish use an initialization vector. Indentical files won't
  produce identical ciphers
- Temp files aren't used during encryption. IF the file has to be decompressed
  however, there is no other way than to use temp files(not true for compression).

Demo Version of SandTiger

This SandTiger demo has the following restrictions

- Only 4 character password limit. The full version has up to 256 chars
- Only CAST-128 cipher is avail in the demo. The full version includes CAST-128, Blowfish and Diamond2.
- You are limited to MD5 hash, whilst the full version has MD5,SHA-1, and RIPEM-160

System Requirements

Windows95 or Windows NT 3.5/4.0.
minimum 386+ processor, pentium 100+ recommended
The amount of memory required depends on how many files you are encrypting

Diamond2 Block Cipher *by Michael Paul Johnson*

The Diamond2 Block Cipher is a royalty-free, symmetric-key encryption algorithm based on a combination of nonlinear functions. This block cipher may be implemented in hardware or software. Diamond uses a block size of 128 bits and a variable length key. A faster variant of Diamond2, called Diamond2 Lite, uses a block size of 64 bits.

Introduction

General symmetric key block ciphers have numerous applications in computer security, communications security, detection of data tampering, and creation of message digests for authentication purposes. The longer any one such algorithm is used, and the more use it gets, the greater the incentive to break it, and the greater the probability that methods will be devised to break the algorithm. For example Michael J. Wiener has shown that breaking DES is within the capabilities of many nations and corporations [1]. This sort of reduction in the relative security of DES was anticipated several years ago. One proposed solution is the International Data Encryption Algorithm (IDEA) cipher [2], which was described in [3] and [4] as the Improved Proposed Encryption Standard (IPES). Another one is the MPJ Encryption Algorithm [5], which evolved to the Diamond2 Block Cipher. In the field of cryptography, it is good to have many strong block ciphers available.

Design of Diamond2

Diamond2 was designed to be strong enough to provide security for the foreseeable future. It was also designed to be easy to generate keys for, and to be practical to implement in hardware, software, or in a hybrid implementation

Strength

Three major factors influence the strength of a block cipher: (1) key length (and key setup time), (2) block size, and (3) resistance of the algorithm to attacks other than brute force (such as differential cryptanalysis) [3] [6]. The key length is variable to allow you to select your own trade-off between security and volume of keying material needed. The block size is chosen to make brute force attacks using precomputed tables require an obviously intractable amount of data storage.

Diamond2 uses a variable length key. The use of at least a key with at least 128 bits of entropy is recommended for long term protection of very sensitive data, as a hedge against the possibility of computing power increasing by several orders of magnitudes in the coming years.

The block size for the Diamond2 Block Cipher is fixed at 128 bits, because larger block sizes are unlikely to make any practical difference in security, and because this is a convenient binary multiple (16 bytes). Diamond2 Lite has a block size of 64 bits because this is good enough for most applications, and because it allows a much faster total avalanche effect and greater software speed than the 128-bit block size.

The problem of making sure that there is no known attack that is more efficient than brute force is much more difficult than simply selecting sizes for keys and blocks. This is attempted by creating a composite function of simpler nonlinear functions in such a way that the internal intermediate results cannot be solved for and such that there is a strong dependence of every output bit on every input bit and every key bit. Another important consideration is that the author and inventor keep up with significant developments in cryptanalysis. This last requirement is only partially met, in that a large percentage of significant cryptanalysis technology is shrouded in secrecy.

An ideal 128 bit block cipher would use a **z** bit key to select one of 2**z** functions from the set of all one to one and onto functions that map one input block of 128 bits to one output block of 128 bits. Ideally, these 2**z** functions would be the most nonlinear and difficult to analyze functions out of the (2128)! possible functions. In practice, the key selects one of 2**z** functions from an arbitrary selection of possible functions.

The use of purely nonlinear functions makes a large portion of mathematical tools ineffective for cryptanalysis. The tools that remain are defeated by ensuring adequate complexity in terms of time and memory requirements that solutions are not practical.

Legal Issues, Diamond2 Copyrights & Legal Notices

The Diamond2 and Diamond2 Lite Block Ciphers may be used for any legal purpose without payment of royalties to the inventor or his employer, however the names "Diamond2 Block Cipher" and "Diamond2 Lite Block Cipher" are Trade Marks owned by the inventor, and may not be used in connection with any algorithm that does not comply with the reference implementation given herein. The Diamond2 Block Cipher is the same as the Diamond, MPJ and MPJ2 Encryption Algorithms, with the exception of the key expansion algorithm. Some governments may restrict the use, publication, or export of strong encryption technology.

Diamond2 and Diamond2 Lite are Trade Marks of Michael Paul Johnson. Other trade marks mentioned herein belong to their owners and are mentioned for identification purposes only.

Some cryptographic, cryptanalytic, and key management software and technical data is subject to export controls and other legal restrictions. Contact competent legal authority for more information. It is your responsibility to comply with all currently valid laws and treaties that apply to you. Do not use this software or technical data for any illegal activity.

As far as is permitted by law, permission is hereby granted to copy and use the copyrighted portions of this distribution for any legal use, provided that you don't misrepresent its source or modify the documentation without my permission.

CRC.H, CRC.CPP, DIAMOND.H, and DIAMOND.CPP are in the Public Domain.

Conclusion

Diamond2 and Diamond2 Lite are two of several alternatives to the aging and now relatively insecure DES algorithm. Source code for a software implementation of Diamond2 in C is available in the USA and Canada on the Colorado Catacombs BBS at 303-772-1062 in the file DLOCK2.ZIP or on the Internet as ftp://ftp.csn.net/mpj/I_will_not_export/crypto_???????/file/dlock2.zip, where the ??????? is revealed in ftp://ftp.csn.net/mpj/README. Comments, questions, and reports of possible weaknesses should be sent to the author at one of the following. I recommend that you ask me if any weaknesses have been found in the Diamond2 Block Cipher before using it in any critical applications.

NOTE: The contact below is for reaching the creator of the Diamond2 cipher, and has nothing to do with support for SandTiger. It's purely for those interested in obtaining information about the Diamond2 Cipher(tm).

BBS: 303-772-1062
Internet mail: m.p.johnson@ieee.org, mpj@csn.net, or mpj@netcom.com
CompuServe: 71331,2332

Advantages of Diamond2

No one has broken Diamond2 (or its predecessors, MPJ and MPJ2), yet. See the US$314.16 challenge.

The block chaining mode is time-tested and well respected.

Complete source code is included for your examination and to facilitate porting to other platforms.(click here how to go about getting it)

The cipher text is the same size as the plain text.

It is free.(only the cipher)

You are free to use the algorithms and/or code in this distribution to incorporate encryption into your own applications, without payment of royalties or delays.(applies only to the original diamond2 cipher.)

Diamond2 and Diamond2 Lite, when incorporated into a system that weakens the effective key length and resists modification by the user to the satisfaction of the NSA, may be exportable. Contact the Department of State and the NSA for details and additional requirements.

The encryption is too strong to be generally exportable. There are no intentional weaknesses or trap doors in the algorithm or the program.

Disadvantages of Diamond2

Key management is all manual.

The ciphertext reveals the size of the plain text (but not its contents).

No 7-bit ASCII armoring (uuencoding or radix-64 encoding) is built in for EMAIL purposes -- use another utility to do that.

The encryption is too strong to be exportable without a lot of hassles and controls on the destinations.

If you forget your passphrase, your encrypted data is as good as gone. I can't get it back, no matter how important it was.

Diamond2 Challenge -- The author of Diamond2 Block Cipher, Mike Johnson

THE US$314.16 CHALLENGE

OK, US$314.16 is not enough to pay for the time it would take to do serious cryptanalysis of the Diamond2 Encryption Algorithm, but it is enough to prove that data encrypted with DLOCK2 is secure against the average hacker. The file 31416.ENC was encrypted with DLOCK2.EXE. If you are the first person to (1) decrypt 31416.ENC and (2) follow the instructions in the decrypted file to claim your prize before noon UTC, 20 September 2000, then you will get US$314.16 of my hard-earned money. To claim this prize, you must reveal how you deciphered the ciphertext. You must also not break the law (including any currently valid export laws) in the process of earning this prize. If the ciphertext is not broken, I get to keep my money.

The plain text that 31416.ENC was encoded from is plain, uncompressed, 7-bit ASCII with both CR and LF at the ends of lines. It contains English text, including instructions on how to claim the prize and contact the author.

THE FAIR CHALLENGE

The US$314.16 challenge given above is probably unfair, unless I really goofed badly in the implementation of DLOCK2 or the invention of the Diamond2 Encryption Algorithm. On the other hand, if you find what you think is a weakness or error in either DLOCK2 or Diamond2 (other than the disadvantages listed above), please let me know. There is no cash reward for such information, but I will use the information to help improve the encryption programs that I write.

Blowfish Block Cipher

- Block Cipher: 64 bit block
- Variable Key Length: 32 bits to 448bits
- Designed by Bruce Schneier
- Much faster than DES or IDEA
- Unpatented and royalty free
- Free source code available

Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)
Bruce Schneier Counterpane Systems, 730 Fair Oaks Ave, Oak Park, IL 60302
schneier@winternet.com

Abstract

Blowfish, a new secret-key block cipher, is proposed. It is a Feistel network, iterating a simple encryption function 16 times. The block size is 64 bits, and the key can be any length up to 448 bits. Although there is a complex initialization phase required before any encryption can take place, the actual encryption of data is very efficient on large microprocessors.

The cryptographic community needs to provide the world with a new encryption standard. DES [16], the workhorse encryption algorithm for the past fifteen years, is nearing the end of its useful life. Its 56-bit key size is vulnerable to a brute-force attack [22], and recent advances in differential cryptanalysis [1] and linear cryptanalysis [10] indicate that DES is vulnerable to other attacks as well.

Many of the other unbroken algorithms in the literatureþKhufu [11,12], REDOC II [2,23, 20], and IDEA [7,8,9]þare protected by patents. RC2 and RC4, approved for export with a small key size, are proprietary [18]. GOST [6], a Soviet government algorithm, is specified without the S-boxes. The U.S. government is moving towards secret algorithms, such as the Skipjack algorithm in the Clipper and Capstone chips [17].

If the world is to have a secure, unpatented, and freely- available encryption algorithm by the turn of the century, we need to develop several candidate encryption algorithms now. These algorithms can then be subjected to years of public scrutiny and cryptanalysis. Then, the hope is that one or more candidate algorithms will survive this process, and can eventually become a new standard.

The implementation of Blowfish used in SANDTIGER was from the SSLeay-0.8.0 library. Please read the Legal Notices and Copyrights.

Blowfish Copyrights

This package contains a Blowfish implementation written by Eric Young (eay@cryptsoft.com).

This product includes software developed by Eric Young (eay@cryptsoft.com)

Frequently Asked Questions

What's with the default password?

The default password was used to quickly test out the app without having to type in the passphrase everytime I encrypted something. I left it in for people just trying it out to save them from the same tedious job of typing in the password anytime they encrypted something.

Why do I get the dialog window saying "xxx.xxx - archive not saved"?

basically when you create a new archive, Sandtiger puts them in a temp file and considers it temporary until you decide to save it. if you double click on an unsaved archive or try to extract from an archive that hasn't been saved, you get that dialog box.

How do I know that my copy has been registered?

The Register menuitem will be disabled if your copy is registered

Can I use the unregistered version to decrypt files encrypted with the registered version?

Well, kind of. If your password was less than or equal to 4 chars, you would be able to decrypt files encrypted with the registered version. Future versions will allow you full keylength during decryption.

Scramble filenames?

By default SandTiger encrypts files, but not their filenames. Anybody who knows his way around computers can use a hex editor to see the filenames. Scrambling filenames would prevent anybody from getting a "clue" as to what kind of files are in the archive.

What about me? will I be able to see the encrypted filenames?

Sure. But only after you enter the correct password.

Can the archive password be different from the password of the encrypted files?

Yup. Remeber that SandTiger can have many passwords in a Single Archive.

SandTiger User Manual

Getting Started

Basic Operation

SandTigers purpose is simple. To encrypt files and nothing else.

ENCRYPTING
1. You will be prompted for a password. This password will be used to encrypt only the files you are about to select. You can enter a different password each time you choose to encrypt a single or group of files.

2. Select a directory containing the files you want to encrypt. You can set a mask to encrypt only a type of file. Eg, entering *.doc in the mask box will encrypt only file extension ending with ".doc".

3. After making your selection, press the OK button to encrypt. You'll see a progress bar telling you the progress of the encryption.

4. The encrypted files are listed in the main Listview window.

5. You may choose to execute steps 1-4 until you have encrypted all the files you want to encypt, each time using a different or the same password.

5. To save, choose the save or close option in the menu. You will be prompted for a filename and your encrypted archive is saved.

DECRYPTING
1. Open an archive by choosing the FILE->OPEN menu or dragging the file from explorer and dropping it into the sandtiger main window.

2. Select the files you'ld like to decrypt (use CTRL + LEFT_CLICK or SHIFT+LEFT_CLICK) to select multiple files.

3. Choose Action->Extract option in the menu

4. You'll be prompted for a password, and then later to select a directory to decrypt your files. NOTE: The files will fail to decrypt and the operation aborted if your password is incorrect.

Toolbar Operation

The toolbar places you a button away from important operations performed in SandTiger. Initially when SandTiger starts up, some buttons will be grayed indicating the operation is not available. This feature is used throughout Encrypt-It to conform to the Microsoft Windows object - action concept. For example, you must first create an archive, by choosing NEW before you can add encrypted files. If an archive hasn't been created, both the ADD and EXTRACT buttons will be disabled.

New Archive

This is the first operation you must perform to start encrypting files. It creates a temporary archive which you encrypt your files into. Most of the buttons will be enabled by now.

Open Archive

To open an existing archive and perform file operations on it

Save Archive

By default, a temporary file is created when you click on the NEW button. This temporary is marked for deletion immediately after SandTiger exists, so you must save your newly created archives. SandTiger will prompt you before it exists on the status of this temporary file

Close Archive

Saves the archive and closes it. You can use this before you create a new archive, although pressing the NEW button automatically closes an archive and opens a new one

Add

Starts a process where you enter a password, select files, which are encrypted with your password and saved in the archive. The filenames appear in the main window.

Extract

Extracts selected files from an archive

Delete

Deletes selected files from an archive. Files deleted aren't physically deleted until the archive is closed.

Find

Searches for a filenames, or filenames in an archive

Find Next

Finds the next matching filename. Press F3 to use a shortcut

Help

Opens this help file

Exit

Exists SandTiger. Sandtiger will prompt you if it needs to save the archive

Setting Default Options

Click on View->Options in the file menu to get to the options property Sheet. It is separated into two tabs, Encryption options and General Options.

Encryption Tab

Encryption

You can set the default encryption algorithm here. Choose between CAST-128, Blowfish and Diamond2

Hash

The security of your password depends on the security of these hash algorithms. Each of these hashes will generate a unique ID of your password+random bytes. At this moment SHA-1 is preferred to the MD5 and RIPEMD-160, although MD5 has been in existence for a long time and hasn't been broken yet.

Mode

Only CBC mode is used in Version 2.0

General Tab

General Options

Scramble Filenames

By default SandTiger encrypts files, but not their filenames. Anybody who knows his way around computers can use a hex editor to see the filenames. Scrambling filenames would prevent anybody from getting a "clue" as to what kind of files are in the archive.

Compress files

Compressing files can drastically reduce the space these archives takes. Compression also enhances encryption by removing redundant bytes from a file making it more secure. The bad side to compression is, it's quite slow and increases the time it takes to encrypt files. It's advantageous to use compression with text files or highly compressible files. It can save you a lot of space

Show Extended Infomation

Shows additional columns in the Main Window. You can view details about a file, eg whether it is compressed "CPR" or uncompressed "RAW", and the encryption type etc.

Disable Default Password

The default password is LOCK, and was used for internal purposes to avoid entering a filename each time the application was tested. Do not use it

Beep After Operation

Beeps after a file operation. Encryption or Decryption

Save Config on Exit

Saves environmental information on exit

Display file association icons

Displays associated Icons of files in the main window. A quick way to tell what type of file, the filename represents

Update List Window

Not Used anymore. Redundant

Tutorial

Diamond uses a block size of 128 bits and a variable length key. The use of at least a key with at least 128 bits of entropy is recommended for long term protection of very sensitive data, as a hedge against the possibility of computing power increasing by several orders of magnitudes in the coming years.

Blowfish, a new secret-key block cipher, is proposed. It is a Feistel network, iterating a simple encryption function 16 times. The block size is 64 bits, and the key can be any length up to 448 bits.

Message Digest 5 algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. Your password is the message of arbitrary length, which is added to 8 randomly generated bytes and then processed by MD5 to produce the 128-bit "fingerprint" or "message digest". Well tested, but considered to be too old.

Secure Hash Algorithm takes as input a message of arbitrary length and produces as output a 160-bit "fingerprint" or "message digest" of the input. Your password is the message of arbitrary length, which is added to 8 randomly generated bytes and then processed by SHA-1 to produce the 160-bit "fingerprint" or "message digest". Generally thought of as the replacement for the ageing MD5. Produces a longer fingerprint too.

RIPEmd-160 algorithm takes as input a message of arbitrary length and produces as output a 160-bit "fingerprint" or "message digest" of the input. Your password is the message of arbitrary length, which is added to 8 randomly generated bytes and then processed by RIPEmd to produce the 160-bit "fingerprint" or "message digest". Generally agreed to be as good as SHA-1.

CAST uses a 64-bit block size which is the same as the Data Encryption Standard (DES). The CAST-128 algorithm supports variable key lengths, anywhere from 40 bits to 128 bits in length

CBC - Cipher Block Chaining Mode

ECB - Electronic Code Book Mode

OFB - Output FeedBack Mode

Normally it only takes a snooper to look at the filenames contained in an archive to see what the fuss is all about. Scrambling filenames allows SandTiger to encrypt all the file and folder name strings. You supply the password and this password is used to encrypt all the names. You MUST enter a valid password or Sandtiger would refuse to open it. You are in effect "locking" the archive.

Allows you to see various flags to each individual files. ie. The type of encryption, hash, if it's compressed and so on.

Although slow, compression greatly reduces the size of the archive. Compression technology replaces repeating bytes with a single character and hence, the compression. Note: Using compression technology on already compressed files (eg, pictures) is not very useful. Usually text files compress very well.

Your password is usually used by most ciphers to generate a key that is used to actually encrypt a file. A "good" password has a length of 8 bytes or longer, isn't the name of your girl friend, "QWERTY" or some other common phrase. A nonsensical password mixed with numbers is secure and preferrable.

Since the password is masked, you should reenter it in this edit box to confirm it was typed correctly.

Checking this box will instruct SandTiger not to show this dialog box for the remainder of this session. By doing so, you intend to use the same password for the remainder of this session. To use a new password, you will have to restart SandTiger.

Folder recursion allows you to encrypt all folders below the folder that you selected. Using folder recursion along with Save Folder Infomation allow you to maintain your directory structure. Very useful if you are moving a group of files from one computer to another.

By default SandTiger only saves the filenames of encrypted files. Using this option allows you to maintain the folder/sub folder the file was located in. For example, sandtiger will encrypt a file in "C:\HELLO\aboutme.txt" as "aboutme.txt". Checking this option will instruct sandtiger to save it as "\HELLO\aboutme.txt", the "\HELLO\" part can later be created as a folder if you wish.

Select a folder and files to work with

Enables the extraction of files whilst maintaining the directory structure

By default SandTiger will not overwrite files that already exist. Use this option to overwrite existing files.

Your use of this software indicates your acceptance of this license agreement and warranty.

Refusing to accept this license agreement will terminate the program. Your use of this software indicates your acceptance of this license agreement and warranty.

Please enter your username and serial number to register this software. Entering "shareware" and
"evaluation" for the serial number will let you try out this application.

Please enter your username and serial number to register this software. Entering "shareware" for USERNAME and "evaluation" for the SERIAL NUMBER will let you try out this application.

You should carefully read the following terms and conditions before using this software. Your use of this software indicates your acceptance of this license agreement and warranty.

Initialization vector

One of the problems with encrypting such things as files in specific formats (i.e., that of a word processor, email, etc.) is that there is a high degree of predictability about the first bytes of the message. This could be used to break the encrypted message easier than by brute force. In ciphers where one block of data is used to influence the ciphertext of the next (such as CBC), a random block of data is encrypted and used as the first block of the encrypted message, resulting in a less predictable ciphertext message. This random block is known as the initialization vector. The decryption process also performs the function of removing the first block, resulting in the original plaintext.