ScramDisk for Linux

Version 2.1-0

User Guide and Technical Documentation

Hans-Ulrich Jüttner

February 13, 2011

Contents

1 Introduction

ScramDisk for Linux, or *Sd4L* for short, is an on-the-fly encryption system which hides complete file systems within encrypted regular files called containers. *ScramDisk for Linux* also encrypts partitions on a hard disk or storage media such as USB sticks or floppy disks entirely as devices. Even complete hard disks with several partitions can be encrypted entirely as a single container. If configured accordingly it is container-compatible with Scramdisk for Windows written by Shaun Hollingworth which, however, is no longer beeing actively developed. Moreover, *ScramDisk for Linux 2.1* can open and create *TrueCrypt* containers of the versions 4, 5, 6 and 7. The *ScramDisk for Linux* project is hosted by SourceForge.net and its home page is "*http://sd4l.sourceforge.net/*". The homepage of the *TrueCrypt* Project is "*http://www.truecrypt.org/*". For details on the old Scramdisk for Windows see for instance the World-Wide-Web page "*http://en.wikipedia.org/wiki/Scramdisk*".

For bug reports on *ScramDisk for Linux* you could use the SourceForge.net tracker system from the projects summary page "*http://sourceforge.net/projects/sd4l/*". There are also three mailing lists of the *Sd4L* project. The list <*sd4l-announce@lists.sourceforge.net*> is only used by the Sd4L team to announce new versions of *ScramDisk for Linux*. Subscribe to this list if you want to stay informed about new versions. For support requests on *ScramDisk for Linux* write to the mailing list <*sd4l-user@lists.sourceforge.net*>. Technical and development questions may be discussed on <*sd4l-devel@lists.sourceforge.net*>. You may also send mail to the author <*huj@users.sourceforge.net*>. General questions on Scramdisk might be better placed in the news group <*alt.security.scramdisk*>.

ScramDisk for Linux comprises a kernel driver ***scramdisk.ko*** (***scramdisk.o*** for systems with kernel 2.4.x), the graphical user interface ***scramdisk*** and five small utilities ***sdcreate***, ***sdchange***, ***sdmount***, ***sdreformat*** and ***sdumount*** as well as the program ***sdhelper*** which is for internal use by the kernel driver only. After a container has been "mounted" any data could be read from or written to the file system inside the container transparently with help of the driver *scramdiks.ko*. By "unmounting" the container the file system becomes inaccessible again. This can be done by the *Mount* and *Unmount* buttons of the graphical user interface or by ***sdmount*** and ***sdumount***. Automatical unmounting after a period of inactivity given on mounting is also possible.

New encrypted containers to be used by ScramDisk can be created using the *Create* button or with ***sdcreate***. All ciphers of the Windows Scramdisk software and some additional ciphers like *AES* are supported except for *Idea*, wich is patented, *Tea* with 16 rounds and *Misty1*. Moreover, *ScramDisk for Linux* supports the additional digests *Ripemd160*, *SHA256*, *SHA512* and *Whirlpool*. If windows container compatibility is checked, the created container can be opened by the Windows Scramdisk as well. In this case a digest or a cipher not supported by the Windows Scramdisk is not accepted and the container is formatted *msdos*. On the other hand, any container created with Scramdisk for Windows can be opened with ScramDisk for Linux if it does not use *Idea*, *Tea* with 16 rounds or *Misty1*. The passphrase of a container can be changed later using the *Change* button or with ***sdchange***. *TrueCrypt* containers created with *ScramDisk for Linux* always can be opened with those verisions of *TrueCrypt* which are equal or higher than the chosen one. Vice versa, containers created with *TrueCrypt* from version 4.1 to version 7 can be opened by *ScramDisk for Linux 2.1* if no keyfiles have been used which ScramDisk doesn't handle. A container may also be reformatted from ScramDisk to TrueCrypt format with the *Reformat* button or with ***sdreformat***. A file based container will thereby decrease by 9728 Bytes in size. Only the formats of TrueCrypt 4 and 5 are created by this process.

As usual for Linux systems, there is also a man page besides the present documentation. It may be called by any of the commands

```
man scramdisk
man sdchange
man sdcreate
man sdmount
man sdreformat
man sdumount
```

in a shell. This man page serves the purpose of a quick and succinct information.

ScramDisk for Linux is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

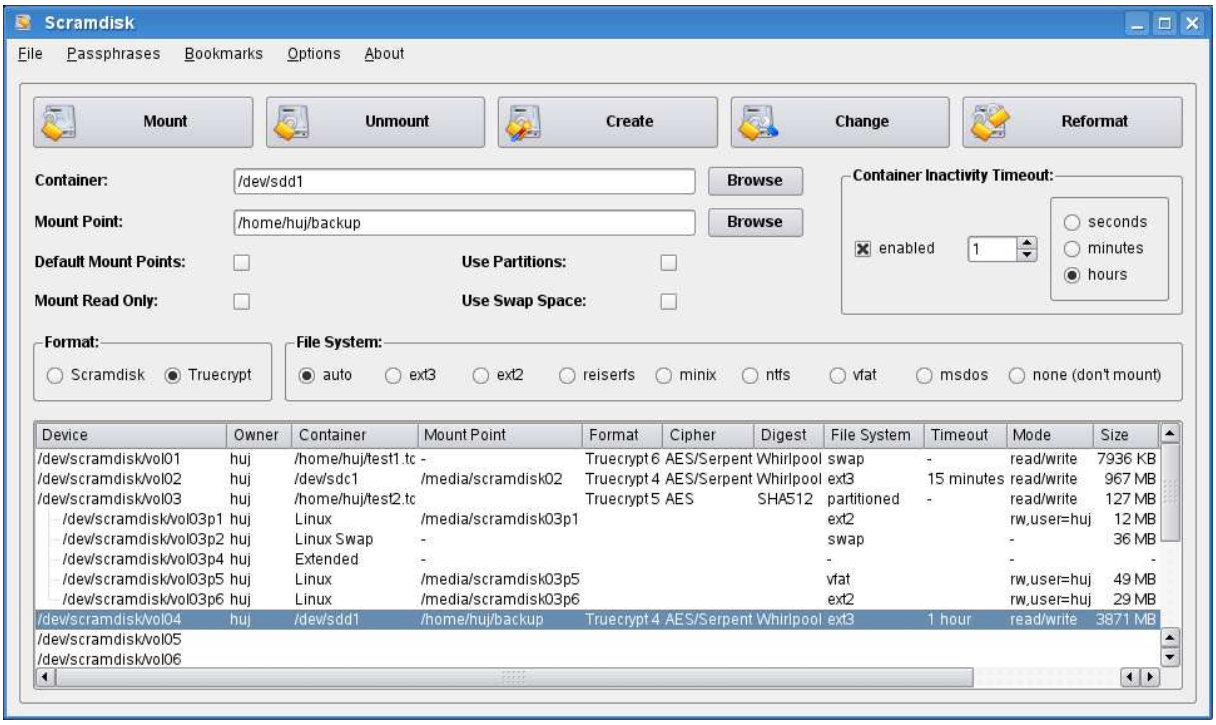
ScramDisk for Linux is distributed in the hope that it will be useful, but **without any warranty**; without even the implied warranty of **merchantability** or **fitness for a particular purpose**. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with *ScramDisk for Linux*—see the file COPYING. If not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307, USA.

I wish to thank Sam Simpson, who encouraged the development of *ScramDisk for Linux* for many years and greatly helped organizing the first steps. My special thanks are due to Andy Jeffries, who made most valuable first steps towards the kernel driver. However, the code has been completely revised and the errors are, of course, mine. I also like to thank Hannes Beinert for initializing and maintaining the SourceForge.net project and the SourceForge.net team for hosting the project.

2 The Graphical User Interface

The program ***scramdisk*** at present has english and german translations. The language is chosen at start time from the configuration file or the locale set in the `LANG` variable in the users environment. An existing configuration entry for the language overwrites the value of the locale. When started in english language, it presents the following graphical control center which allows all functions of *ScramDisk for Linux* to be performed.



The five buttons on top of the window start the functions of *ScramDisk for Linux*. They are also accessible with the *File* menu from the menu bar or by the hotkeys *Ctrl+M*, *Ctrl+U*, *Ctrl+R*, *Ctrl+H* and *Ctrl+F* respectively. All mounted containers are shown on the list at the bottom of the window. However, the list will not be updated if a container is mounted or unmounted by another process, either *sdmount*, *sdumount* or a second *scramdisk* program, while *scramdisk* is running. For containers mounted by other users on the system, the details are omitted. When starting the graphical user interface anew, the list of containers is read from the kernel driver. Therefore, it is complete again after that. Not contained in that list are, however, those TrueCrypt containers, which have been mounted by TrueCrypt's own software for Linux. This is so, because the ScramDisk kernel driver can only administer the containers which have been mounted by himself.

2.1 Mounting Containers

In order to mount a container first choose the container file in the line labeled "*Container*". This can be done by directly typing the files path name into the line or by browsing the file system with the *Browse* button at the end of the line. Similarly you must choose a mount point in the line labeled "*Mount Point*" below the container line. On Linux mounted file systems are not administrated by drive letters as on Windows systems. Instead they become part of the hierarchy of directories. The existing—and preferably empty—directory under which the new file system will be presented is called the mount point. By checking the check box "*Default Mount Points*" it is possible to let ScramDisk itself choose the mount point. Then it will create and use directories */media/scramdisk01*, */media/scramdisk02* and so on as mount points.

The radio buttons labeled "*File System*" can be left on the default "*auto*" which means that the container will be probed for the right file system. If the radio button labeled "*none (don't mount)*" is checked, the container will be decrypted but the file system inside will not be mounted when the *Mount* button is pressed. This only makes sense for special administrative purposes. There is also a check box "*Mount Read Only*". If it is checked the container will be opened for reading only and all write accesses to it will be denied. The check boxes "*Use Partitions*" and "*Use Swap Space*" are to be checked if the container comprises partitions or if it is formatted as swap space and shall be used for swapping out areas of the main memory. This also works together for particular partitions within the container which are formatted as swap space. In the case of partitioned containers new directories `p1`, `p2` etc. below the chosen directory will be created and used as mount points for the partitions of the container. If "*Default Mount Points*" has been checked at the same time, directories `/media/scramdisk01p1` etc. will be created and used.

You can open TrueCrypt containers with *ScramDisk for Linux 2.1* also. For this purpose check the *TrueCrypt* button in the button group on the left hand side. For ScramDisk containers the button *ScramDisk* in this group must be checked.

When you press the *Mount* button you will be prompted for the passphrase if it hasn't been entered beforehand by the *Passphrases* menu. The passphrase dialog for ScramDisk containers presents four lines for the passphrase for up to 39 characters each. For TrueCrypt containers it has only one line for up to 64 characters. The passphrase must be entered there in the same way it has been set when the container was created or when the passphrase was changed.

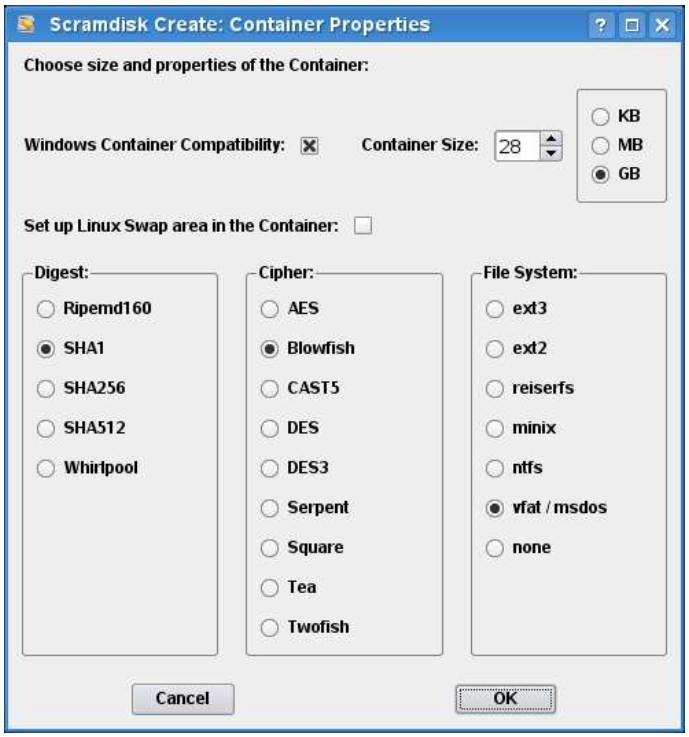
2.2 Unmounting Containers

In order to unmount a container simply mark it in the list at the bottom of the main window and press the *Unmount* button. You can also mark several or even all containers in the list and unmount them at once by pressing the *Unmount* button. Of course you can only unmount the containers you have mounted yourself. For these containers your user name is presented in the *Owner* column of the list. Other containers are ignored by the unmount procedure.

A container may also be automatically unmounted. This behavior must be enabled on mounting the container with the check box "*enabled*" in the group labeled "*Container Inactivity Timeout*". Then, the unmount will happen after the container has been inactive for the time configured behind this check box. Here, inactivity means that no read or write operation has taken place on the container for the given period of time. This timeout, which may be different for several mounted containers, is displayed with other properties of the mounted containers in the list at the bottom of the window. A hyphen in the timeout column means that there is no timeout for the container.

2.3 Creating new Containers

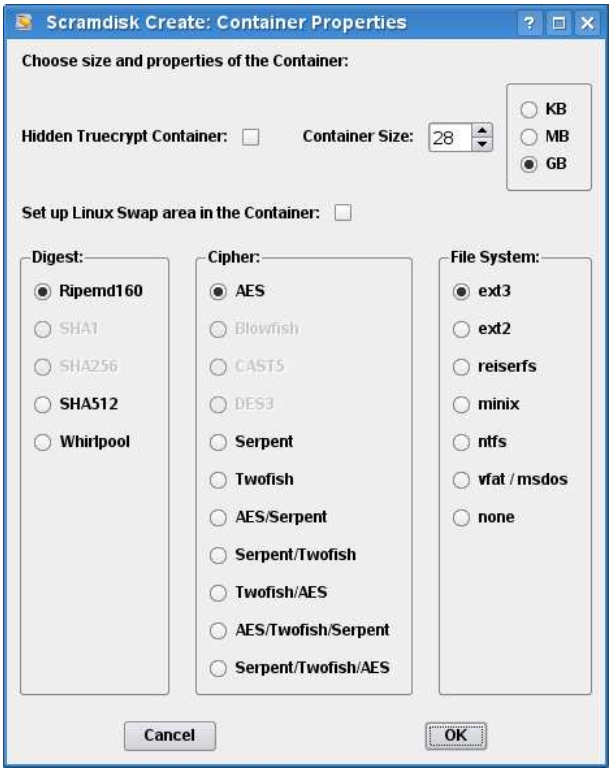
New containers can be created by pressing the *Create* button. First of all, you get a dialog where you have to choose between ScramDisk and TrueCrypt 4, 5, 6 or 7 container format. After that, you get a file dialog. Select the containers file name and directory with it. You may as well pick a block device here which is usually located under the directory `/dev`. By that a partition on a hard disk or a storage medium will be encrypted as container. By subsequently pressing the *Save* button in the file dialog you get the following window for the ScramDisk container format:



Specify the properties of the container you want to create. If the box for "*Windows Container Compatibility*" is checked, it is guaranteed that the created container can be opened by the Windows Scramdisk as well. In this case an error message is displayed if any other conflicting option is chosen. The size of the container can be specified in KB, MB or GB. For the encryption there is a choice between five digest and nine cipher algorithms. If you have no idea about this stuff you can safely keep the defaults *SHA1* and *Blowfish*. Six different common Linux and Windows file systems are supported. For a Windows compatible container "*vfat / msdos*" must be chosen. The option "*none*" is normally not recommended. If it is chosen you have to create the file system by yourself.

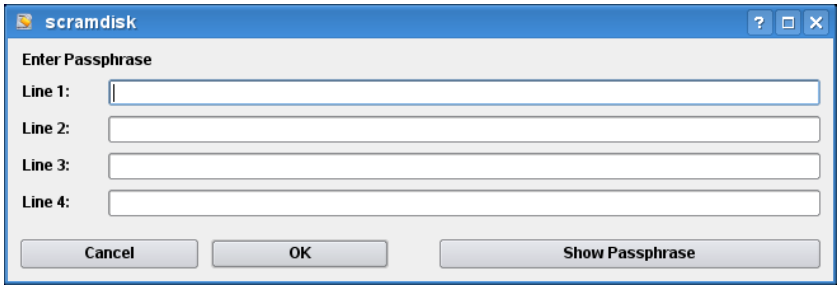
If the check box "*Set up Linux Swap area in the Container*" is checked, such a swap area will create within the new container. Afterwards the container could be used for swapping and paging of main memory. By this the swapped out memory will be encrypted. Since a container couldn't be used as file system and as swap area simultaneously, the file system options will be greyed out if this box is checked.

If, at the beginning, the TrueCrypt container format was chosen, you instead get the following window:

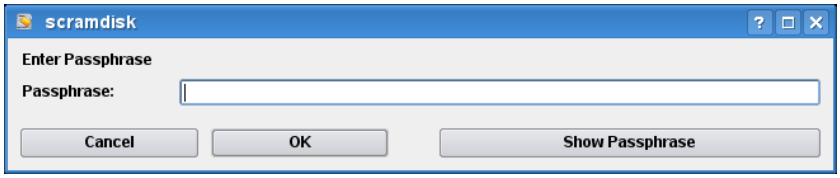


The box for "*Hidden TrueCrypt Container*" can only be checked if an existing TrueCrypt container was selected to create the hidden volume within (see the documentation on "*http://www.truecrypt.org/*" for the meaning of a hidden TrueCrypt volume). In that case you are prompted for the passphrase of the existing container if it wasn't given beforehand. For TrueCrypt containers you have the choice between five digest and eleven cipher algorithms. According to the version of the TrueCrypt format some digests or ciphers which are not available in the chosen version are greyed out. The remaining properties are the same as for ScramDisk containers.

After finishing the property dialog with the *OK* button you are prompted for a passphrase which for ScramDisk containers looks as follows:



The passphrase dialog presents four lines for up to 39 characters each. You are encouraged to use all of them for a stronger passphrase. They are concatenated to make up the final passphrase of the new container. By hitting the *Show Passphrase* button the passphrase is presented in the clear instead of by asterisks. When the *OK* button has been pressed the passphrase must be verified by entering it a second time. This is a safeguard against a typo which you probably could not reproduce later. For TrueCrypt containers the passphrase dialog only has one line of up to 64 characters:



Next you are usually asked to move the mouse which is needed to gather the random data for building the container. Progress bars show the state of three distinct stages of the build process. Finally, you are informed by a message box when the container has been created. TrueCrypt containers need less random data than ScramDisk containers. Therefore, the mouse moving is not so tedious and time-consuming for them. Sometimes they are not even necessary at all if enough random data are already available.

2.4 Changing Passphrases and Repairing Containers

The passphrase of a container can be changed using the *Change* button. By doing so, one can sometimes even recover a container with minor damages. As in the procedure of creating containers, you first will be asked to choose the container format, ScramDisk or TrueCrypt. After that, a file dialog will be presented by which the container must be chosen. Then you are asked for the old passphrase.

Next the question is posed wether to use the backup block in order to repair the container if it is damaged. You normally say *No* to this question. But if this doesn't work and you assume that the container has been damaged you may try to say *Yes*. The versions 4 and 5 of TrueCrypt doesn't have such a backup block. Therefore, using the backup block will fail in that case. After this has been answered you are queried for the old passphrase. When it has been entered correctly you can give a new passphrase which must be verified subsequently just as in the procedure of creating a container. If it could be verified a message box informs you that the passphrase has been changed successfully.

For ScramDisk, the backup block is a triply encrypted copy of the header block which is decrypted with the passphrase. If that block is corrupted by at least one bit which was flipped it wouldn't decrypt. But the same information can be gained by decrypting the backup block. If the backup block itself is not corrupted we exactly have the situation where we can recover from the damage. For TrueCrypt containers of versions 6 and 7 the backup block is encrypted with a different salt value in lieu of the triple encryption used by ScramDisk.

The change of the passphrase is, however, not sufficient to stop someone who knows the old passphrase and had access to the container in the past from opening the container in the future. If he had copied the old header of the container—this is the first 10 KB of a ScramDisk container or the first 512 bytes of a TrueCrypt container—he or she just has to copy this old header back on the new container in order to open it with the old passphrase again. Therefore, if this attack scenario must be considered, you better create a completely new container and copy the contents of the old container to the new one. This procedure prevents the delineated attack.

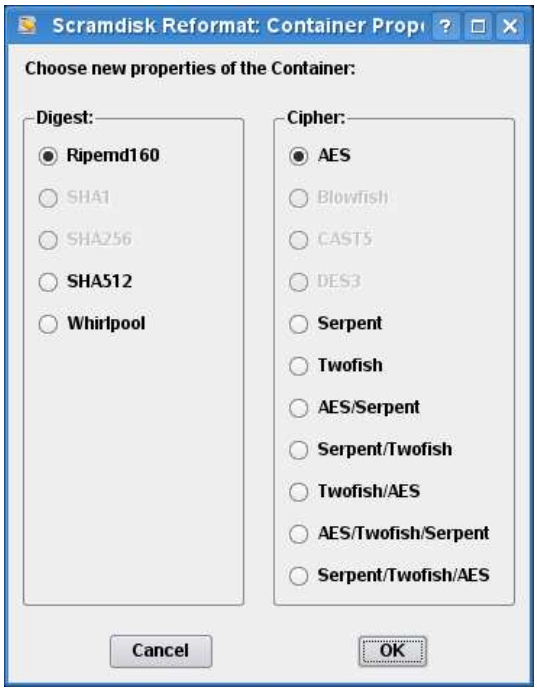
2.5 Reformatting Containers

The format of a container may be changed from ScramDisk to TrueCrypt format with the *Reformat* button. Only this direction of format change is supported since the TrueCrypt format with the LRW mode[1] or the XTS mode which is an enhanced variant of the LRW mode is the more modern one. Moreover, while a container file decreases by 9728 Bytes in this format change, it would increase by the same amount in an reversed format change which would lead to problems where the memory on the data carrier is already exhausted. Since the TrueCrypt 6 and 7

[1] Named after Liskov, Rivest and Wagner, see below.

formats in contrast to that of versions 4 and 5 take even more space than the ScramDisk format, reformatting to this format isn't possible.

After clicking *Reformat* a file dialog is presented by which the container to be reformatted must be chosen. After that you first have to give the passphrase of the old ScramDisk format of the container. Next the new passphrase to be used with the TrueCrypt format must be given and has to be verified just as in creating a container or changing the passphrase. If this has been done successfully the following dialog asks for the new properties of the container:



In contrast to the creation of a TrueCrypt container only digest and cipher have to be chosen here. Size and file system of the container are determined by the container as it is so far. A hidden TrueCrypt container cannot be created by this process. Decryption and re-encryption of the whole container is necessary for reformatting and will be done subsequently. For large containers, this may take quite some time. Therefore, a progress bar is presented. It is not advisable to cancel this process because this would lead to data loss. For that reason a warning message is popped up if the *Cancel* button was hit and you are asked if you really want to cancel the reformatting process.

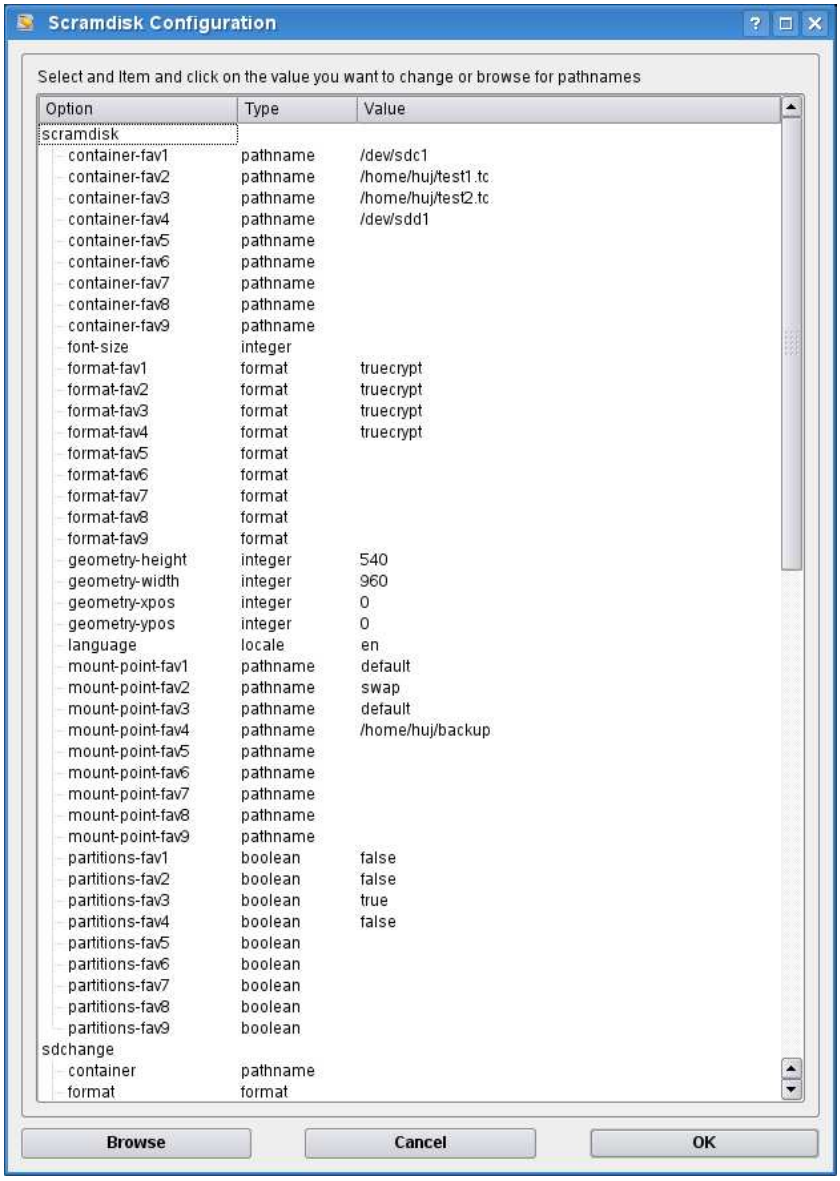
2.6 Bookmarks of ScramDisk

The *Bookmarks* in the menu bar present nine different shortcuts for favorite quadruples of container, mount point, format and the information wether the container is partitioned. If you have specified such a favorite beforehand (see the next section), it may be accessed by klicking on it in this menu or by the corresponding hotkey (*Ctrl+1*, . . ., *Ctrl+9*). Then, the lines for container and mount point are set to the values you have configured for that favorite and the appropriate radio button for the format is selected. Application of these shortcuts speeds up switching between a number of containers you are using simultaneously.

For the mount point in such a favorite the special values "*default*" and "*swap*" may be chosen. In the first case the container will be mounted on one of the default mount points; in the second case it will be used as swap space.

2.7 Configuring ScramDisk

ScramDisk may be configured individually. For this purpose choose *Options* in the menu bar and then *Configure*. You get the following editable table which is also accessible via the hotkey *Ctrl+C*:



In the first column the options are listed in a tree structure. They are grouped in the six categories *scramdisk*, *sdcreate*, *sdchange*, *sdmount*, *sdreformat* and *sdumount* corresponding to the names of the graphical user interface and the command line utilities. The graphical user interface *scramdisk* also utilizes the options for *sdcreate*, *sdmount* and *sdumount* when it performs the corresponding functions.

The second column lists the type for each option. The type on an option specifies the range of values which are possible for the option. ScramDisk handles the following types:

Table 1: Types of configuration variables

type	possible values
boolean	`true`, `false`
integer	any positive number
locale	a country code scramdisk has been translated to, at present `en` and `de` are supported
format	a container format, `scramdisk` or `truecrypt` respectively `truecrypt4`, `truecrypt5`, `truecrypt6` or `truecrypt7`
cipher	`aes`, `blowfish`, `cast5`, `des`, `des3`, `serpent`, `square`, `tea`, `twofish`, `aes/serpent`, `serpent/twofish`, `twofish/aes`, `aes/twofish/serpent`, `serpent/twofish/aes`
digest	`ripemd160`, `sha1`, `sha256`, `sha512`, `whirlpool`
filesystem	`ext2`, `ext3`, `reiserfs`, `minix`, `ntfs`, `vfat`, `msdos`
pathname	any text which can be the name of a file or directory

The third column lists the current value of the option which is configured in your home directory in the file `.scramdisk/scramdiskrc`. If this file does not yet exist, all value fields are left empty. In order to change the value of an option first select the option then click on its value field and type in the new value or use the *Browse* button for pathnames. The new values are rejected immediately if they are not compatible with their types. Finally you confirm your changes with the *OK* button or dismiss them with the *Cancel* button.

The size of the font used by the graphical user interface of ScramDisk is adjustable here. This change, just like a change of the language and other options which have already been processed, only take effect after a restart of the program.

Below *Options* there is, furthermore, the item *Save Geometry* by which the current size and position of the main window of ScramDisk are saved in the configuration file. This action may also be executed with the hotkey *Ctrl+G*. Thereafter, ScramDisk will start with the geometry of the state current when saving last.

2.8 Online Documentation

There is an online documentation in the graphical user interface. In order to access it choose *About* from the menu bar and then *Documentation* or use the hotkey *Ctrl+D*. You get a simple browser presenting some small pages which explain the features and handling of the graphical user interface. These pages are all installed on your computer with the *ScramDisk for Linux* software. So you don't need any internet connection.

The version number of ScramDisk may be displayed with *Version* from the *About* menu or with the hotkey *Ctrl+V*. This displays at the same time the version nummer of the Linux kernel for which the ScramDisk for Linux package at hand has been built. This may be used to verify that the package is compatible with the running kernel.

3 Command Line Tools

3.1 Mounting and Unmounting Containers

In order to make the files within the container accessible the container must be mounted which requires two steps. The first step in mounting loads the master key and some other material into the kernel driver and decrypts it with the passphrase supplied by the user. One of the block devices `/dev/scramdisk/vol01` to `/dev/scramdisk/vol15` is assigned to the container to be mounted. If there is no more free device available an error is returned and the container will not be mounted.

In the second step the file system is mounted at the mount point specified by the user. An entry is added to the system configuration file `/etc/mtab` which describes all currently mounted file systems. These two steps are carried out by the utility *sdmount* in conjunction with the kernel driver *scramdisk.ko* (*scramdisk.o* for kernel 2.4.x).

The utility *sdumount* first unmounts the file system and removes the corresponding entry from `/etc/mtab`. Secondly it instructs the kernel driver to erase the passphrase and master key of the specified container from its memory and to free the assigned block device again. Only the user who mounted the container and the super user *root* are allowed to unmount this container.

The five command line utilities *sdmount*, *sdumount*, *sdcreate*, *sdchange* and *sdreformat* all are configured in the same configuration file. It is devided into distinct sections for this purpose. There is a system wide default for this configuration file "`/etc/scramdisk/scramdiskrc`" and—possibly—user specific configuration files "`.scramdisk/scramdiskrc`" in the home directories of users. A totally different configuration file may, however, be specified on the command line by the option `-cfg` <*file name*>. In case of conflicting configurations the file given on the command line has priority over the user specific configuration file which in turn has priority over the system wide configuration file. Any specification given directly on the command line overrides all configured values.

3.1.1 Command line parameters of sdmount

Except for `-cfg`, `-help` and `-version` there is a one to one correspondence of command line and configuration parameters for *sdmount* as well as the other ScramDisk utilities. Their names are identical to the command line parameters with omission of the leading hyphen. The configuration parameters for *sdmount* are looked up from the section of the configuration file headed by "`[sdmount]`". Table 2 lists all command line options with all possible parameters and their meaning for *sdmount*. Some command line options also have short forms. The container file must always be given to *sdmount* on command line or in a configuration file with the exception of the option "`-info`".

Table 2: Command line parameters of sdmount

option	short form	parameters	default value	meaning
`-cfg`		file name	`$HOME/.scramdisk/scramdiskrc`	specific configuration file to be used in place of the system wide configuration file `/etc/scramdisk/scramdiskrc`
`-container`	`-c`	file or device name		the container to be mounted

-format		scramdisk, truecrypt	truecrypt	format of the container to be mounted (truecrypt4, truecrypt5, truecrypt6 and truecrypt7 are synonymic with truecrypt; the truecrypt version is determined from the container)
-mount-point	-mnt	directory name		the directory where the file system is mounted
-default-mount	-defmnt		false	mount the container on a directory automatically chosen from /media/scramdisk01 ... /media/scramdisk15 (these directories will be created by sdmount and removed by sdumount)
-read-only	-ro		false	mount the container read only, if a regular container file is not writable by the user it is always mounted read only
-partitions	-part		false	mount the partitions from inside the container (for this purpose directories p1 ... p15 will be created by sdmount and removed by sdumount)
-swap			false	use the container for paging and swapping (if -partitions has been selected, a swap partition will be used for paging and swapping)
-no-fs			false	don't mount the filesystem, only load the encryption key into the kernel
-fstype	-t	auto, ext2, ext3, reiserfs, minix, ntfs, vfat, msdos	auto	type of the filesystem to be mounted, if auto is specified, the actual type will be determined automatically
-timeout	no timeout	positive number		set a timeout for the container in seconds; when no read or write operations have taken place on the container for the given period of time the container will be unmounted automatically
-show-pass			false	show the passphrase while typing (instead of asterisks)
-info	-i		false	print information on all mounted containers
-version	-v			print the ScramDisk version number
-help	-h			print a help message on program usage

3.1.2 Command line parameters of sdumount

There are less configuration parameters for *sdumount*. At least the container or the mount point must be given on command line or in a configuration file except for the option "`-all`" or "`umount-all`" in the configuration file. The section for *sdumount* in the configuration file is headed by "`[sdumount]`". Table 3 lists all command line options with their possible parameters and their meaning for *sdumount*.

Table 3: Command line parameters of sdumount

option	short form	parameters	default value	meaning
`-cfg`		file name	`$HOME/.scramdisk/scramdiskrc`	specific configuration file to be used in place of the system wide configuration file `/etc/scramdisk/scramdiskrc`
`-container`	`-c`	file or device name		the container to be unmounted
`-mount-point`	`-mnt`	directory name		the directory where the file system is mounted
`-no-fs`			`false`	don't unmount a filesystem, only erase the encryption key from the kernel which makes sense if no filesystem has been mounted
`-swap`			`false`	stop using the container for paging and swapping
`-all`	`-a`		`false`	unmount all containers mounted by the user
`-brutal`	`-b`		`false`	unmount the container even when there are open files or directories within it
`-fusr-command`		command name	`/bin/fuser`	command to check for processes accessing a file system (on most systems this is `/sbin/fuser`)
`-version`	`-v`			print the ScramDisk version number
`-help`	`-h`			print a help message on program usage

3.2 Creating, changing and reformatting Containers

The command line utility *sdcreate* generates ScramDisk containers subject to a variety of options given as parameters on the command line or, alternatively, read from a configuration file. With the tool *sdchange* the passphrase of an existing container can be changed. Of course, the old passphrase has to be supplied to *sdchange* beforehand. In this process a partly damaged container header may also be restored in case of a good backup block in the header. For this the command line option `-use-backup` must be specified.

Scramdisk for Windows within a disk sector uses cipher block chaining for encryption and decryption. For ciphers which operate on 128-bit blocks[2] this cipher block chaining is applied only to the first half of each block. Because this seems somewhat artificial, ScramDisk for Linux normally does not follow this procedure. However, there is a

[2]Scramdisk for Windows supports only the 128-bit block cipher *square*.

configuration parameter and the command line switch `-win-compatible` by wich the behaviour of Scramdisk for Windows can be enforced. If a container with a 128-bit block cipher created by *sdcreate* shall be accessible with Scramdisk for Windows `-win-compatible` must be specified. This option also disables ciphers and digests which are not supported by Scramdisk for Windows.

3.2.1 Command line parameters of sdcreate

The configuration parameters for `sdcreate` are looked up from the section of the configuration file headed by "`[sdcreate]`". Table 4 lists all command line options with all possible parameters and their meaning for *sdcreate*. Out of the command line parameters `-kb`, `-mb` and `-gb` at most one may be specified. If neither of them is specified there must be a specification in the configuration file. The container file must always be given to *sdcreate*. The given size is, however, ignored if the container is a block device. In this case the partition or storage medium is completely filled with the container.

Table 4: Command line parameters of sdcreate

option	**short form**	**parameters**	**default value**	**meaning**
`-cfg`		file name	`$HOME/.scramdisk/scramdiskrc`	specific configuration file to be used in place of the system wide configuration file `/etc/scramdisk/scramdiskrc`
`-container`	`-c`	file or device name		the container to be created
`-format`		`scramdisk`, `truecrypt4`, `truecrypt5`, `truecrypt6`, `truecrypt7`	`truecrypt7`	format of the container to be created
`-digest`		`ripemd160`, `sha1`, `sha256`, `sha512`, `whirlpool`	`sha1`	algorithm for hashing the passphrase, `sha256` isn't supported by all TrueCrypt containers, `sha512` not by TrueCrypt 4, `sha1` not by TrueCrypt 5, 6 and 7
`-cipher`		`aes`, `blowfish`, `cast5`, `des`, `des3`, `serpent`, `square`, `tea`, `twofish` for ScramDisk and `aes`, `blowfish`, `cast5`, `des3`, `serpent`, `twofish`, `aes/serpent`, `serpent/twofish`, `twofish/aes`, `aes/twofish/serpent`, `serpent/twofish/aes` for TrueCrypt	`blowfish`	algorithm for encryption and decryption, `blowfish`, `cast5` and `des3`, aren't supported by TrueCrypt 5, 6 and 7
`-kb`		positive number		size of the created container in KB
`-mb`		positive number		size of the created container in MB
`-gb`		positive number		size of the created container in GB
`-no-fs`			`false`	don't create a filesystem

-fstype	-t	ext2, ext3, reiserfs, minix, ntfs, vfat, msdos	ext2	type of the filesystem to be created
-mkfs-command		command name	/sbin/mkfs	command to be called for creating the filesystem
-swap			false	set up a Linux swap area in the container
-mkswap-command		command name	/sbin/mkswap	command to be called for creating the swap area
-win-compatible			false	containers shall be compatible with the Windows scramdisk even for ciphers with 128-bit block size
-hidden			false	create a hidden TrueCrypt container within an existing container
-random	-rnd	device name	/dev/random	device from which random bytes are read
-urandom	-urnd	device name	/dev/urandom	device from which random bytes can be read without interruption
-show-pass			false	show the passphrase while typing (instead of asterisks)
-version	-v			print the ScramDisk version number
-help	-h			print a help message on program usage

3.2.2 Command line parameters of sdchange

The configuration parameters for sdchange are looked up from the section of the configuration file headed by "[sdchange]". Table 5 lists all command line options with all possible parameters and their meaning for *sdchange*. The container file must always be given to *sdchange*.

Table 5: Command line parameters of sdchange

option	**short form**	**parameters**	**default value**	**meaning**
-cfg		file name	$HOME/.scramdisk/scramdiskrc	specific configuration file to be used in place of the system wide configuration file /etc/scramdisk/scramdiskrc
-container	-c	file or device name		the container to be changed
-format		scramdisk, truecrypt	truecrypt	format of the container to be changed
-use-backup			false	use the backup header block to retrieve the key and update the main header block if the key opens the container
-show-pass			false	show the passphrase while typing (instead of asterisks)

-version	-v			print the ScramDisk version number
-help	-h			print a help message on program usage

3.2.3 Command line parameters of sdreformat

The configuration parameters for sdreformat are looked up from the section of the configuration file headed by "[sdreformat]". Table 6 lists all command line options with all possible parameters and their meaning for *sdreformat*. The container file must always be given to *sdreformat*.

Table 6: Command line parameters of sdreformat

option	**short form**	**parameters**	**default value**	**meaning**
-cfg		file name	$HOME/.scramdisk/scramdiskrc	specific configuration file to be used in place of the system wide configuration file /etc/scramdisk/scramdiskrc
-container	-c	file or device name		the container to be reformatted
-format		truecrypt4, truecrypt5	truecrypt5	format of the new container
-digest		ripemd160, sha1, sha512, whirlpool	sha1	algorithm for hashing the passphrase in the new format
-cipher		aes, blowfish, cast5, des3, serpent, twofish, aes/serpent, serpent/twofish, twofish/aes, aes/twofish/serpent, serpent/twofish/aes	aes	algorithm for encryption and decryption in the new format, blowfish, cast5 and des3, aren't supported by TrueCrypt 5
-random	-rnd	device name	/dev/random	device from which random bytes are read
-show-pass			false	show the passphrase while typing (instead of asterisks)
-rename			false	rename an .svl ending of the container to .tc
-version	-v			print the ScramDisk version number
-help	-h			print a help message on program usage

4 The Kernel Driver

The kernel driver *scramdisk.ko* (*scramdisk.o* for kernel 2.4.x) is loaded as module to the Linux kernel by the command `insmod`. It has to be build for the exact kernel version of the kernel it is loaded to. Therefore, different *ScramDisk for Linux* packages are provided for several different Linux distributions. If scramdisk has been installed the driver will be loaded automatically when the system boots up. It makes block devices available for up to 15 simultaneously mounted scramdisk containers[3]. The major number of the device driver has been chosen to be 125. Besides read and write access it provides the special *ioctl*-commands `SCRAMDISK_IOC_MOUNT` and `SCRAMDISK_IOC_UMOUNT` by which the utilities *sdmount* and *sdumount* as well as the graphical user interface mount or unmount containers. By the *ioctl*-command `SCRAMDISK_IOC_INQUIRE` the kernel can be asked for information on mounted containers, it is used by the graphical user interface and by *sdmount* with the option `-info`/`-i` for information on mounted containers.

If supplied with the correct passphrase once by the user with the utility *sdmount* or the graphical user interface, the driver decrypts read and encrypts write operations to the mounted container transparently. This is done only for the user who supplied the passphrase by mounting the container and for the system administrator *root*. However, after the file system within the container has been mounted by the user or the system administrator, any user on the computer may access it as long as it is allowed by the rights of the file system. If asked to unmount a container, the driver erases passphrase and master key and the data within the container become completely inaccessible again.

The driver probes for the actual cipher and digest algorithm for the particular container during the mount process. For ScramDisk containers this is done by decrypting three special blocks within the header with all candidate pairs of implemented cipher and digest algorithms. If any two of the three decrypted blocks show up to be identical in comparism the supplied passphrase must be correct and the correct cipher and digest algorithms have been found. If this does not happen the passphrase must be wrong and the container can't be mounted. Usually, all three blocks are the same when decrypted correctly. The third block is provided for the test just in case one of the other two blocks has been damaged.

For TrueCrypt containers the correct decryption with the passphrase is checked for all candidate pairs of cipher and digest algorithms by the string "`TRUE`" at position 64 in the decrypted TrueCrypt header and by a *CRC-32* checksum at position 72. For TrueCrypt 6 and 7 containers there is another checksum at position 252 which is also checked.

After a container has been mounted, the ScramDisk for Linux kernel driver works similarly to the loop device of the Linux kernel. A separate kernel thread is started for each mounted container. It handles all input and output requests for the container. These threads show up in the output of the `ps` command with options `aux` or `fax` as `[scramdisk01]` for the first container, `[scramdisk02]` for the second container and so on.

[3] These are the devices `/dev/scramdisk/vol01` ... `/dev/scramdisk/vol15`.

5 Encryption Scheme and Container Format

ScramDisk as well as TrueCrypt containers consist of a header and the encrypted volume which comprises the file system and the contents of all the files. In both formats header and volume are—without knowledge of the passphrase—indistinguishable from perfect random numbers. In particular, ScramDisk and TrueCrypt containers without knowledge of the passphrase cannot be separated from files or partitions or storage media filled with random numbers.

ScramDisk containers have a 10 KB header at the beginning of the container. The header contains the master key by which the file system is encrypted. Moreover, the header contains a so called *whitening table* with random data which ensures that identical sectors in the container are encrypted differently. Thus, an attacker wouldn't know these blocks are identical when he examines the encrypted container. The master key and the whitening table in turn are encrypted by a key derived from the passphrase using a hash function. Table 7 lists the contents of this header with position, size and encryption in detail.

Table 7: Header of ScramDisk Containers

Position	Size	Encryption	Contents
0	1024	cipher with hash of passphrase	whitening table (random)
1024	1024	cipher with hash of passphrase	master key (random)
2048	2048	cipher with hash of passphrase	unused (random)
4096	2048	cipher with hash of passphrase	unused (0-Bytes)
6144	2048	triply repeated cipher with hash of passphrase	backup of the first 2048 bytes
8192	512	cipher with master key and sector number 0	check block (random)
8704	512	cipher with master key and sector number 1	check block (copy of the first check block)
9216	512	cipher with master key and sector number 2	check block (copy of the first check block)
9728	512	cipher with master key and sector number 3	time and date of the container creation and last passphrase change (12 bytes each at positions 272 and 292 respectively within this block, 0-Bytes elsewhere in this block)

The last 2048 bytes of the header and the following volume are divided into sectors of 512 bytes each which are numbered consecutively beginning with 0. Encryption with the master key depends on that sector number and the whitening table. Every bit of the sector number selects one of two consecutive 32-bit numbers of the whitening table. These numbers, then, are added up with carry to two initial values for the CBC mode[4] of the encryption and two additional initialization vectors which are XORed with the encrypted blocks alternately. By this procedure up to 2^{32} sectors are encrypted differently.

A TrueCrypt container has a header of only 512 bytes at the beginning of the container for versions 4 and 5. It may have a second header for a hidden TrueCrypt container which also has 512 bytes and is located at offset 1536 from the end of the container. TrueCrypt 6 and 7 have two 64 KB headers at the beginning of the container, one for the normal and one for a possible hidden container. Moreover, it has two 64 KB backup headers at the end of the container. The TrueCrypt 4 format uses the modern *LRW* mode[5] which is state of the art and ensures without

[4] In cipher-block chaining (CBC) mode the first block is XORed with the initial value and then encrypted, the second block is XORed with the encryption of the first block and then encrypted and so on.

[5] See Moses Liskov, Ronald L. Rivest and David Wagner: "*Tweakable Block Ciphers*", CRYPTO 2002

a large whitening table that identical sectors or blocks in the container are encrypted differently. TrueCrypt 5, 6 and 7 replace the *LRW* mode with a variant called *XTS* mode. The *XTS* mode avoids a marginal weakness of the *LRW* mode. In TrueCrypt containers the key for decryption of the header is derived by a HMAC method from the passphrase and a salt value. This HMAC method is iterated 1000 or 2000 times depending on the digest which defines the HMAC. For details on the TrueCrypt container format see "*http://www.truecrypt.org/docs/*". The contents of TrueCrypt headers is listed with position, size and encryption in detail in tables 8, 9, 10 and 11 for TrueCrypt 4, 5, 6 and 7 respectively.

Table 8: Header of TrueCrypt 4.x Containers

Position	Size	Encryption	Contents
0	64	not encrypted	salt value (random)
64	4	cipher with HMAC of salt and passphrase	ASCII string "TRUE"
68	2	cipher with HMAC of salt and passphrase	TrueCrypt version number of the format
70	2	cipher with HMAC of salt and passphrase	minimum version nummer of the TrueCrypt program which opens the format
72	4	cipher with HMAC of salt and passphrase	CRC-32 checksum of the (decrypted) bytes 256 to 511 in the header
76	8	cipher with HMAC of salt and passphrase	date and time of container creation
84	8	cipher with HMAC of salt and passphrase	date and time of the last passphrase change
92	8	cipher with HMAC of salt and passphrase	size of the hidden volume or 0-bytes in the first header
100	156	cipher with HMAC of salt and passphrase	unused (0-bytes)
256	32	cipher with HMAC of salt and passphrase	LRW key (random)
288	120	cipher with HMAC of salt and passphrase	master key (random)
408	104	cipher with HMAC of salt and passphrase	unused (0-bytes)

Table 9: Header of TrueCrypt 5 Containers

Position	Size	Encryption	Contents
0	64	not encrypted	salt value (random)
64	4	cipher with HMAC of salt and passphrase	ASCII string "TRUE"
68	2	cipher with HMAC of salt and passphrase	TrueCrypt version number of the format
70	2	cipher with HMAC of salt and passphrase	minimum version nummer of the TrueCrypt program which opens the format
72	4	cipher with HMAC of salt and passphrase	CRC-32 checksum of the (decrypted) bytes 256 to 511 in the header
76	8	cipher with HMAC of salt and passphrase	date and time of container creation

84	8	cipher with HMAC of salt and passphrase	date and time of the last passphrase change
92	8	cipher with HMAC of salt and passphrase	size of the hidden volume or 0-bytes in the first header
100	8	cipher with HMAC of salt and passphrase	size of the volume
108	8	cipher with HMAC of salt and passphrase	byte offset of the encrypted area
116	8	cipher with HMAC of salt and passphrase	size of the encrypted area
124	132	cipher with HMAC of salt and passphrase	unused (0-bytes)
256	*variable*	cipher with HMAC of salt and passphrase	concatenated master key and XTS key (random)

Table 10: Header of TrueCrypt 6 Containers

Position	**Size**	**Encryption**	**Contents**
0	64	not encrypted	salt value (random)
64	4	cipher with HMAC of salt and passphrase	ASCII string "TRUE"
68	2	cipher with HMAC of salt and passphrase	TrueCrypt version number of the format
70	2	cipher with HMAC of salt and passphrase	minimum version nummer of the TrueCrypt program which opens the format
72	4	cipher with HMAC of salt and passphrase	CRC-32 checksum of the (decrypted) bytes 256 to 511 in the header
76	8	cipher with HMAC of salt and passphrase	date and time of container creation, TrueCrypt 6 writes 0-bytes here
84	8	cipher with HMAC of salt and passphrase	date and time of the last passphrase change, TrueCrypt 6 writes 0-bytes here
92	8	cipher with HMAC of salt and passphrase	size of the hidden volume or 0-bytes in the first header
100	8	cipher with HMAC of salt and passphrase	size of the volume
108	8	cipher with HMAC of salt and passphrase	byte offset of the encrypted area
116	8	cipher with HMAC of salt and passphrase	size of the encrypted area
124	4	cipher with HMAC of salt and passphrase	Flag (bit 0 set means system encryption, bits 1 - 31 reserved)
128	124	cipher with HMAC of salt and passphrase	unused (0-bytes)
252	4	cipher with HMAC of salt and passphrase	CRC-32 checksum of the (decrypted) bytes 64 to 251 in the header
256	*variable*	cipher with HMAC of salt and passphrase	concatenated master key and XTS key (random)

512	65024	cipher with HMAC of salt and passphrase	unused (0-bytes), for system encryption this is ommitted
65536	65536	just like in the first 65536 bytes	hidden volume header or random data, built like the first header, for system encryption this is ommitted

Table 11: Header of TrueCrypt 7 Containers

Position	**Size**	**Encryption**	**Contents**
0	64	not encrypted	salt value (random)
64	4	cipher with HMAC of salt and passphrase	ASCII string "TRUE"
68	2	cipher with HMAC of salt and passphrase	TrueCrypt version number of the format
70	2	cipher with HMAC of salt and passphrase	minimum version nummer of the TrueCrypt program which opens the format
72	4	cipher with HMAC of salt and passphrase	CRC-32 checksum of the (decrypted) bytes 256 to 511 in the header
76	8	cipher with HMAC of salt and passphrase	date and time of container creation, TrueCrypt 7 writes 0-bytes here
84	8	cipher with HMAC of salt and passphrase	date and time of the last passphrase change, TrueCrypt 7 writes 0-bytes here
92	8	cipher with HMAC of salt and passphrase	size of the hidden volume or 0-bytes in the first header
100	8	cipher with HMAC of salt and passphrase	size of the volume
108	8	cipher with HMAC of salt and passphrase	byte offset of the master key encrypted area
116	8	cipher with HMAC of salt and passphrase	size of the master key encrypted area
124	4	cipher with HMAC of salt and passphrase	Flag (bit 0 set means system encryption, bits 1 - 31 reserved)
128	4	cipher with HMAC of salt and passphrase	sector size of the volume in bytes
132	120	cipher with HMAC of salt and passphrase	unused (0-bytes)
252	4	cipher with HMAC of salt and passphrase	CRC-32 checksum of the (decrypted) bytes 64 to 251 in the header
256	*variable*	cipher with HMAC of salt and passphrase	concatenated master key and XTS key (random)
512	65024	cipher with HMAC of salt and passphrase	unused (0-bytes), for system encryption this is ommitted
65536	65536	just like in the first 65536 bytes	hidden volume header or random data, built like the first header, for system encryption this is ommitted

6 Building and Installing ScramDisk from Source Code or Package

ScramDisk is written in C and C++ and source code is distributed as GNU-zipped *tar* archive. Some code of the *catacomb* library[6] written by Mark Wooding is included in the subdirectory *crypto*. It has been modified only slightly to be used by the kernel module as well.

Some binary RPM and Debian archives are also provided for a small number of Linux distributions and two hardware architectures. They may be installed, for example, by the command

`rpm -i <`*file-name*`.rpm>`

for RPM packages or

`dpkg -i <`*file-name*`.deb>`

for Debian packages where the archive has to be chosen appropriately for your distribution and processor architecture. Escpecially, make sure that the package matches exactly to your kernel version. The kernel version number is part of the file name of any published package. You can get the version number of your running kernel by the command

```
uname -r
```

All published files in any version of *ScramDisk for Linux* are signed with GnuPG. The signatures are separate `.sig` files. The signature key with user-id "*Ulrich Juettner (ScramDisk for Linux Signature Key) <hans-ulrich.juettner@t-online.de>*" has the following fingerprint:

```
478E BB6C 740F 0C9D D6A2  95EC 51D3 2249 68CE D9BE
```

If you want to build *ScramDisk for Linux* yourself from the source code you need the compiler `gcc` for this task which is usually installed with any Linux distribution. You also need the C++ compiler `g++` which some Linux distributions omit by default.

You need a command line shell as (privileged) user *root* for installation of ScramDisk on your system. On SuSE distributions this can also be done by *YaST*. First make sure you have installed the kernel sources on your system. If not install them from your Linux distribution. Then go to the location of the kernel source and configure them. On a SuSE distribution this can be done by the commands

```
cd /usr/src/linux
make cloneconfig
```

On a Debian or Ubuntu distribution you first have to copy the file `/boot/config-`*Version* as `.config` to the directory `/usr/src/linux`. Here "*Version*" is to be replaced with the version number of your Linux kernel. On Fedora distributions the file to copy is in the directory `/usr/src/linux-`*Version*`/configs`. Then call the following commands on these distributions:

```
cd /usr/src/linux
make oldconfig
```

Afterwards put the *tar*-archive `ScramDisk_2.1-0.tar.gz` in your working directory and unpack it by the command

```
tar xvzf ScramDisk_2.1-0.tar.gz
```

[6] See the World-Wide-Web page "*http://freshmeat.net/projects/catacomb_lib/*".

You also need the Qt library version 3.2 or higher (but not version 4) from Trolltech. Install it with all its developement tools. Then change to the created directory `ScramDisk-2.1` and build and install everything:

```
cd ScramDisk-2.1
make
make install
```

Afterwards you should have the driver installed[7] together with boot scripts which load it as kernel module at boot time. The graphical user interface *scramdisk* and the command line utilities *sdcreate*, *sdchange*, *sdmount*, *sdreformat* and *sdumount* are installed in the directory `/usr/bin`. The program *sdhelper* to be used by the kernel module is installed in the directory `/sbin`. A sample configuration file has been put to `/etc/scramdisk/scramdiskrc`. You may whish to edit this file according to your needs. Alternatively you can put an edited version of this file named "`scramdiskrc`" in a directory "`.scramdisk`" within your normal users home directory.

The *scramdisk* and the command line utilities *sdcreate*, *sdmount* and *sdumount* are installed with the *setuid-bit* set. That means they have root privileges even when executed by a normal user. This is necessary for creating, mounting or unmounting the file system within a container. However, be aware that, therefore, any bug or intentional modification of these utilities is a security problem for your system. A running *scramdisk* withdraws itself the privileges when they are temporarily not needed. This is, however, no absolute safeguard against an attacker who succeeded in manipulating the programm, because it always can retrieve the privileges.

In order to prevent normal users to usurp root privileges by means of *ScramDisk* containers are mounted for normal users in a way such that the execution of programs with *setuid-bit* from within the container is inhibited. Moreover, ScramDisk forbids mounting a container on a mount point to which the user has no write access. This is so, because otherwise a normal user could mount his container over a system directory and thereby replace system files with his own ones.

There is also a target *uninstall* in the `Makefile`. So by doing

```
make uninstall
```

whithin the directory of your source code you can completely remove anything what has been installed by `make install` from ScramDisk for Linux. The kernel sources, of course, must be uninstalled separately. After doing "`make uninstall`" you can remove the directory `scramdisk` with the source code. If you don't need this uninstall feature, the source code may also be removed immediately after installation.

[7] The installation procedure was developed and tested only on Debian, Fedora, SuSE and Ubuntu distributions. On other distributions there might be some modifications.

7 The Configuration File

A typical configuration file looks like the example below.

```
# Example configuration file for ScramDisk for Linux 2.1

[scramdisk]
container-fav1=/media/disk/container1.svl
container-fav2=/home/user/container2.tc
container-fav3=/dev/sdb
font-size=10
format-fav1=scramdisk
format-fav2=truecrypt
format-fav3=truecrypt
geometry-height=500
geometry-width=900
geometry-xpos=0
geometry-ypos=0
language=en
mount-point-fav1=default
mount-point-fav2=swap
mount-point-fav3=/mnt/sdb
partitions-fav1=false
partitions-fav2=false
partitions-fav3=true

[sdchange]
container=container.tc
format=truecrypt
show-pass=false
use-backup=false

[sdcreate]
cipher=aes/twofish/serpent
container=container.tc
digest=sha512
format=truecrypt7
fs-type=ext3
hidden=false
mb=384
mkfs-command=/sbin/mkfs
mkswap-command=/sbin/mkswap
no-fs=false
random=/dev/random
show-pass=false
swap=false
urandom=/dev/urandom
win-compatible=false

[sdmount]
container=/media/disk/container.tc
default-mount=true
format=truecrypt
fs-type=auto
info=false
no-fs=false
partitions=false
read-only=false
show-pass=false
swap=false
timeout=600
```

```
[sdreformat]
cipher=aes/twofish/serpent
container=/media/disk/container.svl
digest=sha512
format=truecrypt5
random=/dev/random
rename=true
show-pass=false

[sdumount]
brutal=false
fusr-command=/bin/fuser
no-fs=false
swap=false
umount-all=true
```

The configuration file is subdivided into sections by lines containing solely a section name in square brackets. Within a section any line of format "`parameter=value`" specifies exactly one configuration parameter. Anything from a token "`#`" to the end of the line is a comment which is ignored by the program. White spaces and empty lines are also ignored by the program. Any utility reads only the section headed by its own name in square brackets. The graphical user interface reads all sections.