DiskProtection

The data encryption package for the Amiga
Version 1.2

Patrick Ohly

1 Legal

1.1 Copyrights

DiskProtection and its documentation	© 1994,95 by Patrick Ohly
xpkFEAL	© 1992,93 by Christian von Roques
FEAL-N	patented by Intellectual Property Department, NTT 1–6 Uchisaiwai–cho, 1–chome, Chiyada–ku 100 Japan
xpkIDEA	© 1992 by André Beck
IDEA, International Data Encryption Algorithm	patented by Ascom–Tech AG, Solothurn Lab Postfach 151 4502 Solothurn, Schweiz
MD5 Message–Digest Algorithm	© 1990, RSA Data Security, Inc.
D3DES	© 1988-92 by Richard Outerbridge
Triton, GUI library	© 1993–1995 by Stefan Zeiger
Icons	© Michael–Wolfgang Hohmann (mickh@iM.Net). They may not be used in other projects without his written permission.

Registered Trademarks are not marked in this documentation!

1.2 Disclaimer

There is no warranty for the user of DiskProtection, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holder or any other party which may distribute the packages provides the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for

a particular purpose. The entire risk as to the quality and performance of the program is with you. Should the program prove defective, you assume the cost of all necessary servicing, repair or correction.

In no event unless required by applicable law or agreed to in writing will the copyright holder or any other party be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the program to operate with any other programs), even if the copyright holder or other party has been advised of the possibility of such damages.

1.3 Licence

DiskProtection is distributed as shareware. The package may be distributed freely, as long as the following conditions are met:

1. Any distribution has to contain all files of this archive without any changes.
2. This package may be freely distributed in mailboxes, InterNet/UseNet, software libraries like the one from Fred Fish, Aminet CD-ROMs and similiar electronic chanels.
3. Disk magazines and other service providers that take additional fees for transmitting the data are not allowed to distribute it without permission of the author!

Is you want to use the program regularly you should send me at least $15, 20,- DM or an equivalent amount in a different currency. Please keep in mind that cheques drawn on foreign banks are difficult to cash. Euro cheques and cash are prefered.

In case the payment of the shareware fee is to difficult or the program is used to rarely to pay, I'm satisfied with an interessting postcard or any other small gift. To register please fill out the file '`Registration`' and send it to me by email or mail.

This version of DiskProtection is not crippled in any way. Therefore the shareware fee is a payment for the goods you already have received, but also gives permission to use any future version. Registered users can be send the most recent version by email. However, this is only possible by mail if the nescessary postage is refunded, too.

New, improved versions will be produced if I'm motivated enough and this concept works. Otherwise it might be possible that future releases will only be fully functional for registered users.

2 Survey

DiskProtection encrypts data totally invisible for the user while writing to disk. All blocks are encrypted, so the directories are protected, too. DiskProtection supports all exec devices, i.e. harddrive partitions, floppy disks, but not the RAM–disk. Any filesystem can be used.

Encryption uses IDEA, FEAL or DES. If the XPK–interface to sublibraries was improved, it would be possible to encrypt with these or other algorithms via XPK. Any number of passwords can be used.

Passwords are requested from the user on demand. They may be made resident in order to avoid requests after a reset. Therefore DiskProtection can be used for mailboxes without permanent control. Alternatively passwords can be hidden in arbitrary files. The computer may be protected from unauthorized use.

There is a preferences program with GUI. The encryption of a partition can be changed anytime: It is automatically encrypted renewedly, so unencrypted data can be converted.

DiskProtection needs an Amiga with at least OS 2.0. It has a graphic user interface designed with Triton V1.3. Locale is supported, catalogs and translations of the complete documentation exist in English and German. The documentation is included in AmigaGuide–, ASCII– and DVI–format and can be displayed as contextsensitive online help.

3 Installation of DiskProtection

DiskProtection is installed with the Commodore installer. You have the choice between two different ways of installation: Everything in an own directory plus some assigns in '`user--startup`' or every part in that directory it is normally expected to be in.

For testing DiskProtection without installing it is enough to set the nescessary assigns with '`MakeAssigns`'. *Note:* When starting DiskProtection, it will always create the files '`ENV[ARC]:diskprot.prefs`'.

DiskProtection is protected with PGP and MD5 checksums against manipulation. The installer script compares the checksums stored in the file '`DiskProtection.readme`' with the files belonging to them automatically. The checksums themselves are protected with a PGP signature that has to be checked manually, though. To do this you start '`PGP`' with this file as its argument. My PGP public key is attached to this file. With the signatures on my public key you can consider if you want to trust it.

Note for users of WB 2.04: The asl.library of these WB version doesn't offer a screenmode requester. DiskProtection uses the reqtools.library instead for this purpose, if installed. Without it you are still able to use all other features of DiskProtection, only selecting a screenmode is disabled.

As reqtools.library is widely spread and not really required in DiskProtection it is not included in this archive. Look at AmiNet for '`/util/libs/ReqTools#?.lha`', if you haven't got it.

4 Concept of DiskProtection

4.1 Approach of DiskProtection

There are a lot of programs encrypting files in many different ways. Most the time the disadvantage is that one can only use the file again in other programs after manually decrypting it or has to use programs capable of reading the file.

One far more comfortable and flexible way to protect one's data is a filesystem or handler, encrypting data or controlling access to files. Solutions of this kind are e.g. MultiUserFilesystem and XFH, that supports the XPK libraries "IDEA" and "FEAL" for encryption. However, the directory and all filenames are still readable without knowing the correct password. So a stranger can still see what is on one's harddrive and gets some hints on the contents of a file. This knowledge could even be used for an attack.

DiskProtection chooses a different approach. Its core is an exec device that encypts every single disk before it is written to floppy disk or harddrive. That way any filesystem can be used. Without this device and the password used to encrypt the disks one only gets an 'Unreadable Disk", without any filename, directory or file contents readable in clear.

This concept has a different disadvantage, though: Preferences can only be set for one unit and therefore only for one partition or disk as a whole. Different encryption in different directories or of different files is not possible.

4.2 Units of the diskprot.devices

The diskprot.device is based on so called units like any other exec device. In this case every *DPUnit* is an integral whole which can be configured independently from the others.

As the diskprot.device does not access hardware itself, but uses other exec devices, too, the most important preference is which device and which of this device's units the DPUnit is to access. This might be "trackdisk.device", unit 0 for '`DF0:`'. Furthermore DiskProtection has to know which disks belong to one DPUnit in order to support harddrive partitions. Removeable media are specially supported. With them any number of media can be converted when changing encryption.

The encryption style includes the algorithm, its mode and the password.

4.3 DOS-Drivers

To write data to a unit, one has to mount a DOS-device accessing this unit. The '`MOUNT`' command requires entries in a '`Mountlist`' or—since Workbench 2.1—single files in the directories '`DEVS:DOSDrivers`' or '`SYS:Storage/DOSDrivers`'. The DOS-devices in '`DEVS:DOSDrivers`' are mounted automatically from 2.1 on from within the startup–sequence, while you have to enter the '`MOUNT`' command for every other DOS–device by hand.

If the directory '`DEVS:DOSDrivers`' exists, then DiskProtection will create DOS–drivers there. Otherwise mount entries are managed in the file '`DEVS:DP-Mountlist`'. You have to add the argument '`FROM DEVS:DP-Mountlist`' to the '`MOUNT`' command then! DiskProtection can move DOS–drivers between '`DEVS:DOSDrivers`' and '`SYS:Storage/DOSDrivers`'. When using the mountlist you have to mount the DOS–devices manually.

DiskProtection itself needs some of the information about a unit from the DOS–driver or mountlist, too: The device uses the entries `SectorSize = BlockSize`, `SectorsPerTrack = BlocksPerTrack`, `BufMemType` and `Flags`, if present.

In order to convert a unit the DOS–drivers of the unit are scanned for `LowCyl`, `HighCyl`, `Surfaces` and `SectorsPerTrack = BlocksPerTrack`. Without this information cannot be converted.

Warning: When manually creating a DOS–driver you should set '`STACKSIZE`' to at least 4000!

4.4 Encryption

DiskProtection can only use encryption algorithms that do not change the length of a data chunk. The data security lies in keeping the password secret, as the algorithms itselves are well known.

There is a interface to the sublibraries of the XPK–packet. Unfortunately the present XPK–concept does not enable sublibraries encrypting a chunk without adding additional information. Therefore xpkIDEA and xpkFEAL were integrated into diskprot.device with permission of the authors. Should XPK be reworked, it could be possible to use sublibraries instead of these internal versions and write your own cryptographic algorithm for DiskProtection.

You find more information on the background of cryptography in Chapter 8 [Basis], page 23.

4.5 Passwords

DiskProtection knows different kinds of passwords, that are used differently. All have in common, though, that *no password text is saved anywhere on disk.* Only the so called hash value of a password is saved (more on this topic in Section 8.1 [Hash Value], page 23). This value is only used to check an entered password. The hash value is not used for en/decryption, for this purpose the correct password is still needed.

There is no “trap door” to find a lost password from the program preferences!

With passwords all characters are allowed and case is significant. A password may be nearly arbitrary long and does not have to be a single word. A string of random characters or a long sentence would be most secure. You should never choose a single word with any relation to your name or person or found in a dictionary. A program looking for correct passwords would check this and variations of it at first. A good compromise is to remember a long sentence, but only using the first letters and punctuation marks. The result is often very random, but easy to remember.

4.5.1 Password Settings

The problem with to many units and one password per unit would be that you were busy with typing passwords after a reset. On the other hand it could be desirable to protect an important unit with an own password. Both is possible with DiskProtection. You can create as many passwords as you like and set the password used for a unit.

A password is requested the first time one of its units is accessed. Then you have an adjustable period to enter the password. The entered password is compared with the saved hash value. Incorrect passwords are rejected immediately. There are unlimited attempts. The request may be cancelled anytime.

Has the period passed without a password being entered, e.g. because you are not at the computer, or the requester was cancelled, then the filesystem accessing the unit will no longer be suspended, but is denied to open the unit.

Without this behaviour an application accessing the filesystem would be suspended, too, wich might not be desirable in certain circumstances. You may set the period to zero, which will leave the password requester open until it is removed manually.

4.5.2 System Password

There is a special password that may not be deleted, but is adjustable, too: the system password. Of course you can configure it as any other password. It is used among others for encryption of the program's preferences file and to protect from unauthorized use of the prefs program. Default text is an empty string, so in the requester a `RETURN` is enough.

4.6 Passwords after a Reset

Entering the passwords is a problem when the computer is running without control and is to continue its work after a reset, as needed for a mailbox. A sysop might turn on the computer and enter all passwords, but after a reset, for example because of a crash, the computer would only boot as long as no password is needed. There are sollutions for this in DiskProtection:

4.6.1 Resident Password

When a password is entered for the first time, e.g. after a cold start, DiskProtection can make it resident in memory and read it from memory after a reset.

A weakness is the possible attack on DiskProtection. Although the passwords are encrypted in memory, the used password must always be the same and with analysing the program code one might find the passwords. In addition the passwords might not survive in memory, for example when a serious crash trashes memory.

Nevertheless this option is still secure, because in order to grab the passwords one must have access to the computer after the passwords were entered and before the computer is turned off.

4.6.2 Hide in File

The disadvantage of the previous option is the unreliability wether the passwords survive a reset or not. If the reliable operation of the computer is more important than the security of the data, then you can choose another method:

A password is read from an arbitrary position in an adjustable file. This file might be hidden in any directory or located on a disk, that might be removed if required. If the file is not present, the normal requester appears. This method might be used in addition to making a password resident. The file is only accessed when the password is not in RAM. The password may extend over several lines, line ends are silently ignored. If the file contents has changed on the next read it will be requested as usual *without warning*.

This method is insecure, though:
The most important drawback is the system itself, as the computer allows access to encrypted data to any user without authentication, as long as the files the passwords are hidden in are present. Without any programming skills anybody can find these files as they are accessed from DiskProtection, e.g. with SnoopDos. Although the whole file is loaded at once, it is possible to find the password itself with analysis of the program code.

4.7 Access Verification

There is one weakness left in the system: After the computer is turned on and the passwords are entered, anyone can use it and read files from secret partitions. After entering the passwords it should be possible to lock the computer and all user input should be ignored.

This realizes DiskProtection with opening a screen on pressing a hotkey and then filtering all user input for actions that might bring this screen to the background. In addition this screen is brought to front automatically in short periods. It is only removed after the system password is entered in the string gadget on this screen. To achieve this, DiskProtection establishs both an input handler and a commodity called '`DPSecurity`'. With '`Exchange`' you can en/disable and activate the access protection screen, too.

You can make DiskProtection activate the access verification screen immediatly when opening diskprot.device the first time. Because it often happens that one leaves the computer for a short while and returns much later than expected due to some delays, the screen can be opened automatically like a screenblanker, in other words after an adjustable space of time passed without user activity.

In contrast to the request of the system password nescessary to gain access to program's prefs file and that way indirectly to encrypted units, the security screen does not prevent the computer from booting! Therefore this option is strongly recommended if you really want to hide the system password in a file (Section 4.6.2 [Hide in File], page 8), because in this combination the computer starts, but grants access only after the system password was entered manually, too.

If the screen cannot be opened, e.g. due to memory lack, one could lock the device itself until the screen can be opened again. So the “screenblanker” and “open on start” options would reliably prevent access to encrypted data. But that way a serious dead lock might occur: Memory is low and the screen cannot be opened. Therefore the device would be locked and an application could not save its data to an encrypted medium. So one could not quit programs in order to free memory $\Rightarrow$ the computer is locked.

Therefore the access verification will simply not be activated if the screen cannot be opened! When you activated it manually, you can react accordingly. When using the screenblanker method there is an additional risk, but the screenblanker method is nothing you should depend on anyway as it activates itself after a certain delay. Finally you can very well infer from your experiences if there is enough memory to open the screen on startup.

Such an access verification is realized by several programs. The drawback is always that—as in DiskProtection, too—these programs can easily be passed by on the Amiga, e.g. by not executing the startup–sequence or booting from floppy. However, with DiskProtection it is certain that no one can avoid this access verification while still using encrypted units, because both access verification and encryption are integrated into diskprot.device and one cannot use one, but avoid the other.

5 DiskProtection, Preferences

With the prefs program DiskProtection you configure all settings of the DiskProtection package. It is named like the whole program package, but may be distinguished from that by its different text style.

5.1 Preferences Concept

DiskProtection is used almost like any other preferences program. Nevertheless there are some differences mentioned here:

5.1.1 Program Start: Arguments, Program Protection

You have to enter the system password every time you start the program. So it is protected from unauthorized use. For this reason DiskProtection may only be used interactively with the graphic user interface. The arguments known from other preferences programs are not supported. There is only one single argument:

```
PubScreen/K
```

It may be given both in the shell and as a tooltype and makes DiskProtection open its windows on the named public screen.

5.1.2 Saving Prefs

If the DiskProtection prefs are not found or were manipulated, you are offered the choice to use the default prefs or cancel loading. When initializing DiskProtection by starting DiskProtection this happens twice, one time the diskprot.device and the other time the prefs program itself loading the prefs. The default password is an empty string.

Very important is also that you keep in mind that all prefs are stored at three different locations:

- ENV[ARC]:DiskProtection.pref

  There are all preferences saved specific to DiskProtection: units, passwords and global prefs. These files are protected from manipulation with checksums and encryption with the system password. The password texts themselves are not saved!
- Units

  Every unit of DiskProtection is encrypted according to the options set for it. If you change this options in '`ENV[ARC]:DiskProtection.pref`', e.g. the password, the whole unit has to be converted in order to still be able to access the files. Formating the unit after activating the new settings has the same effect, but you loose all data.
- '`DEVS:DosDrivers`' and '`SYS:Storage/DosDrivers`' or '`DEVS:DP-Mountlist`'

  There are—as known from AmigaDOS—the DOS–drivers respectively the mount entries the '`MOUNT`' command needs to mount a DOS–device accessing the DPUnits.

You can still change all settings in this single prefs programs. DiskProtection takes care that all three locations are consistent, even if the computer should crash while changing settings or settings are not saved although they were changed.

All changes concerning DOS–drivers are saved immediately. Changes affecting the encryption of a unit are saved **automatically** after converting the corresponding unit.

5.2 Main Window of DiskProtection

5.2.1 Units and Passwords

These both parts are used similar. Both units and passwords are displayed with their names in the corresponding listviews and may be selected there. Units requiring conversion are marked with a prepended asterix (`*`). You change the lists with the buttons below them.

With the button '`New`' a new unit, resp. password is created. With units you are offered to select an already existing DOS–device, whose values are taken for the new unit (s.a. Section 5.5.2 [DOS-Device Selection], page 19). On demand a unit with default values is created, which requires additional manual configuration. For a new password default values are used, too, e.g. `""` as password text.

With ‘`Edit`’ the windows “Edit DPUnit” and “Edit Password” are opened for units, resp. passwords, where the selected entry of the listview in the main window is edited.

Deleting units and passwords is not revertable, but before data is lost, you are always offered the chance to cancel. If a unit is encrypted, you will be asked to decrypt it. Now you can cancel, otherwise the unit is immediately deleted after succesfully converting it.

If you remove a password, then its units are moved to the system password and, if nescessary, converted after confirmation. If a password contains no unit or they can be moved without conversion, the password will be deleted without warning!

5.2.2 System Password and Algorithm

The check button gadget ‘`System Password`’ refers to the password selected in the listview above it and defines which password is used as system password. Deselecting it makes the first password the system password.

Below you choose the algorithm used for encryption of system data with the requester “Algorithm and Mode Selection”.

5.2.3 Access Verification Settings

They are divided into one part for settings belonging to the screen and one part describing the way the access verification is activated:

With ‘`Hotkey`’ you can set a key combination activating the access protection, as common in commodities.

If the access verification is to be activated automatically after a certain period without user activity, then enter the desired period in seconds in ‘`Activate after # secs.`’. Zero disables this feature.

Finally you can activate the screen immediatly at the first invocation of diskprot.device by selecting the option ‘`At Startup`’.

You can set the screenmode and –font used for the accesss protection screen with ASL–requesters.

WB 2.04: Instead of the asl.library the reqtools.library has to be used for the screenmode requester. If it is not installed, this button will be disabled.

Because the access verification screen always stays in front of any other screen, no screenblanker can fulfil its task, as it cannot bring its screen to front. So DiskProtection has an own builtin screenblanker, which reduces the brightness of the frontmost screen to an adjustable percentage (‘`Dimmer`’) after a certain period of time (‘`Blankertime`’). Zero disables this feature.

This blanker does not work for custom screens! Even changing the colors of the workbench or a public screen is illegal and can result in incorrect colors.

5.2.4 Standard Gadgets

The function of these gadgets should be familiar from other preferences programs:

- ‘`Save`’—Save changes permanently in ‘`ENVARC:DiskProtection.prefs`’
- ‘`Use`’—Activate changes, but do not keep them permanently
- ‘`Cancel`’—Quit and discard changes

Keep in mind what was said in Section 5.1 [Preferences Concept], page 11:
Even if you cancel or activate changes only temporary, it is still possible that some of the changes are saved permanently, because they affect DOS–drivers or the encryption of units.

5.2.5 Menus

DiskProtection has all standard menus of a preferences program:

In the menu ‘`Project`’ there are two menu items. With ‘`About ...`’ you get information about your version of DiskProtection. This includes the version and compilation date of the prefs program and the device. ‘`Quit`’ exits the program without saving.

With all three menu items of the menu ‘`Edit`’ you set the current preferences back to certain presets. The limitations mentioned in Section 5.1 [Preferences Concept], page 11 apply here, too. Converted units and changed DOS–drivers are not reset.

- '`Reset to Defaults`':
  Resets *only* the options for access verification and system algorithm to presets built-in the program.
- '`Last Saved`':
  Restores the permanently saved preferences.
- '`Restore`':
  Restores preferences active at program start.

The menu '`Settings`' contains only one item. With '`Create Icons?`' you select if icons are created for DOS–drivers. DiskProtection will use the icon '`def_project.info`' found in '`ENV:Sys`' for that purpose, if present.

5.3 Edit DPUnit

In this window you change all settings for the unit selected in the main window. It is devided into the following parts:

5.3.1 Display of the DPUnit

In this part in the upper left window you are given the internal unit number. This is the number the filesystem has to use to open this unit of the diskprot.device. This number can not be changed and is set by the program in a way that avoids any overlap with existing or previous units. Additionally it is shown if the unit has to be converted, because the settings where changed, but the unit is still not adapted.

The name of the unit you can set here is of more practical use. It is used to identify the unit, e.g. in the listview of the main window. Therefore you should choose a meaningful name. Finally you set if the devices uses removeable media.

5.3.2 Data Protection Settings

In this group the settings concerning data encryption are combined. These settings can always be changed savely without data loss. If you change '`Algorithm`' or '`Password`', you will have to convert the unit after accepting the changes with '`OK`', however (s.a. Section 5.5.1 [Automatic Unit Change], page 18).

With '`Algorithm`' you invoke the requester "Algorithm and Mode Selection" to select the algorithm used for this unit.

'`Password`' sets the password of the unit. Default is the system password. When the new passwword was not already entered, then it is requested now before the unit can be added to it, because otherwise it would not be clear which way the unit is to be encrypted.

5.3.3 Advanced Options

One should edit this options **only** if one knows what he is doing. Normally this options are set to the right values by selecting an existing device on creation of the unit. Only if no device was selected, this options are selectable immediatly. Otherwise '`Changing enabled`' has to be selected first, before the following can be changed:

- In '`Device`' the exec device is given the DPUnit is based on. The device has to be trackdisk.device compatible.
- '`Unit`' is the unit of the selected exec device that is accessed.
- '`FFS Patch Enabled`' may be used with filesystem OFS/FFS/INTL/CACHE versions 2.04 upto 3.1. It must not be used with custom filesystems and exec devices that do not support the command TD_GETGEOMETRY.

  This patch enables the filesystem to recognize different media sizes after a diskchange. It is nescessary to use both HD and DD disks in one DPUnit.

5.3.4 Mountlist Files

Every entry in the listview corresponds to a file in '`DEVS:DOSDrivers`' or '`SYS:Storage/DOSDrivers`' or an entry in '`DEVS:DP-Mountlist`', in other words to a DOS–device. In the '`diskprot`' preferences is saved which DOS-devices where created. With the string gadget '`DOS-Device`' the name is changed.

Functions of the gadgets:

- '`New`' creates a new entry with default values here, too. You definitly have to enter some values manually!
- With '`Copy`' the active entry is copied.

- DOS–drivers are normal ASCII files. So '`Edit`' invokes the Editor set in the enviroment variable *EDITOR*.
- The button '`Remove`' deletes the corresponding file(s)/mountlist entry and the entry in the listview.
- '`Mount`' mounts the DOS–device immediately. For this purpose the settings are used that were already active or the DPunit was created with, in other words **not** the settings only changed within the prefs program.
- With '`Mount at Startup`' you select within DiskProtection, if the DOS–device is mounted at startup of the computer. To achieve this the DOS–driver is moved between '`DEVS:DOSDrivers`' and '`SYS:Storage/DOSDrivers`'. This gadget is disabled when using the DP–Mountlist instead of DOS–drivers.

5.4 Edit Password

In this window you make all selections for the password selected in the main window. The settings here affect units, too, for example changing the passwort text or adding a unit to this password.

Therefore it is possible that you are asked if you want to convert units after accepting the changes (Section 5.5.1 [Automatic Unit Change], page 18). For this purpose DiskProtection creates an internal copy of the password and moves the units there. If you abort while converting, then not converted units are still useable with this second password without any further trouble.

If you want to edit a password and the password text is not yet entered, you will be asked for it, on the one hand to add some further protection and on the other hand to be able to recognize units needing conversion. In addition the password must be entered to be able to hide it in a file.

The string given in '`Password name`' is used to represent the password in listviews or in the password request.

The program aborts the passwort request after '`Waiting period`' seconds without user input automatically. Switched off with zero.

With '`Password`' you invoke a requester to set the text for the current password. For verification you have to enter the password twice, before you can accept it with '`OK`'. That way typos are avoided. This text is **not** saved, but its hash value!

'`Resident`' defines if the password is made resident after entering it (Section 4.6.1 [Resident Password], page 8).

'`Hide`' toggles "Hide in File" of the password. If the password is not entered, the program will ask for it. If no filename is set, then the corresponding string gadget is activated. Alternatively you can invoke a file requester with the gadget beside it.

The file '`Filename`' will be searched for the password if "Hide in File" is enabled. If the password is not found there, you are warned and this feature is disabled automatically.

At the bottom you select the members of the password, at least if the current password is not the system password. For this purpose there are two listviews, '`DPUnits with System Password`' and '`Members of this password`', which show the units that are members of the system password or the current password. '`<-`' and '`->`' move the selected units between the both passwords. This change can cause units to need conversion.

5.5 Requesters of the Prefs program

5.5.1 Changing DPUnit Encryption automatically

After changing settings affecting directly or indirectly the encryption of a unit you are asked, if and when you want to convert the affected unit. Conversion is nescessary to still be able to access the files. Formating the unit after activating the new settings makes the medium useable, too, but you loose all data.

In a requester you are offered the following options, which some of may be disabled depending on the context:

- '`Now`'

  The unit is converted to its new state immediatly. If the conversion is succesful then the changes concerning its encryption will be saved permanently in the file '`ENVARC:DiskProtection.pref`'.
- '`Q-Format`'

  The new settings are always activated and saved permanently in the file '`ENVARC:DiskProtection.pref`'. In addition, DiskProtection tries to format the first valid valid DOS–device of this unit with the option '`QUICK`'. It is mounted first if nescessary.

- '`Later`'

  You may convert the unit later, too. That way you can move a unit to another password and then change this password's text before converting the unit only once. The first time you do not have to convert it, it is enough to convert the data the second time.
- '`Never`'

  The changes are remembered, but not saved yet, and can be cancelled. In contrast to '`Later`' you are not asked to encrypt it again.
- '`Back`'

  Returns to the window you have just left without doing anything.
- '`Cancel`'

  Neither is the unit converted nor are the encryption settings of the unit changed.

Saving or using preferences with units not yet converted causes this requester to pop up for any of these units. However, now you may not preserve changed settings for later conversion. In addition, '`Cancel`' saves the preferences, but without the cancelled changes of unit encryption.

5.5.2 DOS-Device Selection

This requester appears on creation of a new unit. In a listview it offers all mounted DOS–devices suitable for DiskProtection. With '`OK`' a unit with the values of the selected DOS–device is created. '`Default Values`' ignores the selection and creates a new unit with default values as far as reasonable. You definitly have to enter some values manually in this case! '`Cancel`' finally quits the requester without creating a new unit.

5.5.3 Algorithm Method–Requester

With a listview gadget you select one of the currently available data encryption algorithms. Below you are given some information about the selected algorithm. You may set a mode of operation with the slider gadget below, provided that the current algorithm supports different modes. Otherwise the gadget is disabled.

6 Windows/Requesters of the Device

6.1 Password Input

On entering a password there are two input modes: With one you are to enter a certain password. The program knows the hash value (s.a. Section 8.1 [Hash Value], page 23) of this password and complains about incorrect input immediatly. With the other you may enter new password text. In order to avoid typing errors you have to enter the pasword twice before it is accepted.

Depending on the context the window offers the following gadgets:

- Text Display

  Here you finde messages of the program, e.g. the time left to enter the password.
- Password

  You enter the password here. When entering a new password the string gadget is cleared after the first input and you have to repeat the input.
- OK

  Accepts the entered password.
- Cancel

  Quits without returning a valid password.

6.2 Changing Encryption of a DPUnit

This window is opened when a unit is to be converted to a different encryption. The calling program part is told if the conversion took place successfully.

In the upper part you find information about the changes that are to be made. Below is a status display with a fuel gauge, that represents the cylinders already converted, and a text display.

With the options you can set certain aspects of the conversion before starting. Only verifing the written data is currently supported. After selecting this option and inserting a medium, if nescessary, you begin the conversion with '`Start`'. You do not have to convert anything: '`Back`' quits the window as if the conversion was succesful. **Warning:** This makes unconverted media unreadable in this unit, because the encryption does not match the settings! '`Cancel`' leaves the window without making changes.

After starting the conversion the size of the inserted medium is read. That way it is possible to convert both HD and DD disks in random order. After that every single track is read, encrypted differently again and written back. If an error occurs or you select '`Cancel`' a requester dealing with the problem is opened (Section 6.3 [Conversion Errors], page 21).

When the conversion of the current medium is finished it is possible to convert further media with units supporting removeable media. Should conversion be successful with some media but not with all, then carefully consider if you want to quit the window with '`Back`' or '`Cancel`': With the first the converted media, with the second the not converted ones are readable in this unit.

6.3 Conversion Errors

Should an error occur while converting a unit or conversion was aborted then this requester gives an explanation of the problem and offers this options to react:

- '`Undo conversion`'

  All converted tracks are brought back to their previous state and preferences are not changed. The "conversion direction" may be toggled several times.
- '`Repeat`'

  Try again to perform the last action.
- '`Ignore`'

  Continue with the next track. All data of this track are lost! Not selectable with abortion by the user...
- '`Back`'

  Quit conversion as if it was successful. **Warning:** All not converted tracks are lost. Therefore there is an additional warning requester. With removeable media this would have the same effect as '`Cancel`' and is not selectable for this reason.
- '`Cancel`'

  Quits conversion as if you cancelled conversion before starting. **Warning:** All converted tracks are lost. Therefore there is an additional warning requester. With removeable media only the conversion of the current medium is cancelled.

7 DPInit

The DiskProtection system, e.g. the access verification, is activated by invoking the prefs program or mounting a DPUnit. With mounting the corresponding filesystem still has to be started, either by adding '`MOUNT = 1`' to the dosdriver or by accessing the device (e.g. by '`CD DP_xx`'.

Both is sometimes a certain overhead if you only need the access verification. So there is the program DPInit, that does nothing else but opening the diskprot.device. Only at the first start the system password is requested. Afterwards the access verification is available. However, DPUnits still have to be mounted normally. DPInit can be called in the shell or from Workbench and returns error codes on failures.

8 Basis of Cryptography

8.1 Hash Value

A hash value is a number of fixed length that is generated with a hash function of a string with variable length. A hash function fulfilling the following conditions is called a one way hash function:

1. It is easy to calculate the hash value of a string.
2. It is hard to find a string generating a known hash value.
3. It is hard to find a different string generating the same hash value as a known string.

One way hash functions are used to create digital signatures for messages. In DiskProtection the one way hash function MD5 implemented by RSA Data Security Inc. is only used to check entered passwords for correctness. The hash values of all passwords are saved in the prefs file, all encrypted with the system encryption except of the system password's hash value itself.

Even if this hash value should be found by analysing the prefs file format then it is unlikely that the corresponding password is found, too, due to characteristic 2. Even if a password with the same hash value is found, which would be accepted as the correct one after entering it, this does not have to be the password used for encryption and reading encrypted units will still fail.

MD5 itself is used in PGP, too. It creates a 128-bit hash value, long enough to make attacks with brute force hard. It seems to be safe enough for the purpose of DiskProtection.

8.2 Block Encryption

Most encryption algorithms and especially all currently available in DiskProtection encrypt a single chunk of data at once, usually 64 bits. However, there are different ways to use this algorithms.

The easiest way is to encrypt the chunks one by one independently of the others (ECB—Electronic Code Book). Every chunk of plaintext always maps to one chunk of ciphertext. This is a great weakness, as one pair of known plaintext/cipertext would be enough to know the encryption of certain byte series for one key. Additionally single chunks of ciphertext could easily be replaced with ciphertext of another message, which would not be noticed without checksums.

By chaining the chunks with their predecessor these problems are solved. In the CBC1 mode (Cipher Block Chaining) every chunk of plaintext is XORed with the ciphertext of the preceding chunk before encryption. That way encryption of a chunk depends on encryption of any preceding chunk, although it is only trivially slower than ECB.

There is still the problem that diskblocks are encrypted to the same ciphertext up to the position they differ first. This reveals more information than nescessary about the contents of a disk, which might be useful for an attack. DiskProtection has a simple mechanism preventing this: The first longword of any diskblock is xor-ed with value used to access it and then it is encrypted. That way the begining of every encrypted data is different and diskblocks with the same contents result in totally different ciphertexts, too.

Apart from this two modes and a multitude of several variants there are two further modes, encrypting a certain number of bits per encryption of one block (usually less than block size, e.g. one byte) and using chaining, too: OFB and CFB. Both have advantages for encrypting a stream of data, for example the connection between two computers. For encrypting a diskblock there are no advantages.

Even if you have the choice you should always use CBC for DiskProtection, as far as possible. Please mind the special features of a password for IDEA described in '`xpkIDEA.doc`': With a 'normal' password IDEA.76 is safer than IDEA.100!

8.3 The DES Algorithm

DiskProtection includes the "Data Encryption Standard" DES in CBC1 mode. With permission of the author the source code "D3DES" from Richard Outerbridge was used. DES is based on the algorithm "Luzifer" developed by IBM and adopted as a national standard for encryption of unclassified government communication by the NBS (National Bureau of Standards) in the U.S.A. in 1977.

DES encrypts 64-bit blocks and uses a 56-bit key. Encryption and decryption use the same algorithm but a different key schedule. Simply said this algorithm performs 16 rounds of the same operations: replacing and exchanging groups of bits depending on the key. DES is easily implemented in hardware. In contrast software implementations are slower than other algorithms.

History

- 15.05.1973: public request by the NBS to submit proposals for a standard cryptographic algorithm—no suitable submissions
- 27.08.1974: second request
  IBM offers the algorithm “Luzifer” developed in the early 70's for free use. The NSA (National Security Agency) reviewed the proposal and reduced the key length from 128 to 56 bits.
- 1975: publication of the proposal and request for comments
- 1976: two public workshops with lifely discussion about the inner working of the algorithm and the existence of a “trap door”
- 15.01.1977: publication as the “Data Encryption Standard” (DES) and adoption by several standardisation organisations; since then intensive use by industry and banks
- 1987: despite of objections of the NSA DES is reaffirmed as a standard
- 1992: again reaffirmation of the standard, as there are no alternatives available

Security

Much was discussed if the NSA reviewing the algorithm installed a “trap door” in DES apart from only shortening the key length, in order to be able to decrypt DES encrypted data. This suspicion was neglected by a statement of two IBM cryptographers and an investigation by the U.S. senate, although many people remained unconvinced, because the result of the investigation was classified and not published.

Another topic was the key length of 56 bits. There are different calculations of the costs nescessary to build a machine capable to decrypt a message with brute force in a certain period. This considerations all supposed that only big, national organisations like the NSA were able to do so, but due to the technical development such a machine would become cheaper and therefore more likely continuously.

DES has proved to be resistent against several attacks, although there are methods that are more efficient than brute force. Officially DES was only cracked with reasonable effort with a lower number of round than 16. According to Bruce Schneier in 1993 the most efficient attack on DES with 16 rounds, linear cryptoanalysis, still needs 2^43 known plain texts.

8.4 Use of IDEA and FEAL

With permission of the authors slightly changed versions of xpkFEAL and xpkIDEA are included in the current version of the diskprot.device, because it is not possible to use the XPK–concept to access the existing XPK–sublibraries. As the algorithms are the same as in the sublibraries, please refer to the corresponding documentation: IDEA.doc and FEAL.doc.

8.5 The scramble algorithm

Scramble is a very fast encryption algorithm, because it is extremly simple – and therefore totally unsecure. It cannot really protect the contents of your harddrives, but only obscure it. Scramble definitely can and will be broken, if someone really tries to. However it is good enough to make it impossible to read the data at the first glance.

Scramble is based on xoring the data with a value calculated from the password of the DPUnit and some other magic. This value is not the same as the hash value saved in the prefs file, so you still have to know the correct password. I will not tell any details, because the (dubious) security of this algorithm depends partly on the fact that the algorithm itself is unknown.

9 Known Bugs, Tips & Tricks

Requesting the System Password twice

When you do not enter the system password when starting DiskProtection or accessing a unit, you may be asked twice to enter it, even if you cancelled the first request. This may not be fixed in DiskProtection. The password is requested when the diskprot.device is opened. If no system password is entered, opening the device will fail. The operation system itself tries to open the device again now, before it finally gives up and returns control to the application.

DOS Errors with unencrypted DPUnits

When you select no encryption for a DPUnit you can access unencrypted disks in this DOS–device. If the DOS–device normally used to access them is mounted, too, then Amiga–DOS is reported the same volume twice in different devices.

DOS handles this situation badly. For example you are sometimes requested to insert a volume already present. Some CLI commands can still access this volume, others cannot. The same thing happens when you make an exact copy of a disk without changing the date or name and insert original and copy in different drives.

The problem can be avoided if you access a volume always in the same DOS–device, in other words only with the DPUnit or completely without it.

Formating a DPUnit

A medium doesn't have to be converted in order to use it in a DPUnit. It's enough to format it in the DPUnit. The option '`QUICK`' can be used.

If a for example floppy disk is formated that way, DFx: is not notified of the disk content's changes. Therefore a previously normaly formated disk is still present, but at least at the next diskchange or after the command '`DiskChange DFx:`' was issued the changes are recognized.

Getting rid of the original Partition

When you have encrypted a partition, it is still mounted, although it only recognizes a bad disk. With HDToolBox you can get rid of it: Start HDToolBox, partition your drive, select the useless partition, select '`Advanced Options`' and '`Change...`'. Deselect '`Automount this partition`', '`Ok`' (twice), '`Save Changes to Drive`' and exit.

Triton–Prefs for DiskProtection

A little hint: DiskProtection will look a bit nicer if you select something different from the default grey as background pattern for all windows with '`Images`' in the Triton prefs program, e.g. '`Shine/Background`'.

10 History

N: New features
C: Changed
B: Bugfix

- Release 1.0, 25.06.95
  (DiskProtection V1.0, diskprot.device V1.0)
  first public release
- Release 1.0b, 22.07.95
  (DiskProtection V1.1, diskprot.device V1.1)

  B: now works with OS < 2.1 as promised: The program failed when locale.library was not available, although it does not depend upon it. Stupid me ;-)
  B: In a mountlist "Flags" is now set to the correct value.
  C: cosmetic changes
  N: From now on there's a check for the preferences program's version, because a minimum version is required for use with a certain version of the diskprot.device. DiskProtection 1.0 does not perform this check, nevertheless you may not use this version anymore!
- Release 1.0c, 02.08.95
  (DiskProtection V1.1, diskprot.device V1.2)

  B: conversion works with more devices, e.g. A2091 (reading drive geometry was buggy)
- Release 1.0d, 03.08.95
  (DiskProtection V1.2, diskprot.device V1.3)

  B: TD_GETGEOMETRY is only used with trackdisk.device and carddisk.device ⇒ conversion works with all controllers that crash on unknown commands, too.
  B: Programs opening unit 0 could crash (e.g. SysInfo with the SCSI option), because unit 0 is only an internal unit. Fixed.
  N: patch to enable FFS to recognize different disk sizes
  N: Automatically created DPUnit names are unique now.
- Release 1.0e, 08.08.95
  (DiskProtection V1.3, diskprot.device V1.3)

  B: TD_ADDCHANGEINT locked diskprot.device's IO with trackdisk.device ⇒ AFS didn't work. Fixed.
  N: DiskProtection offers a quick format instead of conversion, too.

- Release 1.0f, 09.08.95
  (DiskProtection V1.4, diskprot.device V1.4)

  N: Only one DiskProtection prefs program may run at the same time.
  B: A msg port was not deleted.
  C: rewrote dosdriver inhibiting: some potential bugs removed
  C: minor code cleanup

- Release 1.0g, 21.08.95
  (DiskProtection V1.5, diskprot.device V1.5)

  C: Diskblock encryption: From now on the first longword of every block is xor–ed with the block offset before encryption to make the ciphertext of blocks with the same contents different. This is slightly faster (up to 4%) than the old method of prepending the offset. That is still suported, so disks and partitions with older versions of DiskProtection can be used without changes. However, to use the faster method you have to decrypt the old DPUnit, delete it and create a new DPUnit.
  B: Resident passwords didn't work with OS2.04. Fixed.
  B: Changing a password's text was buggy and could sometimes lead to unpredictable results. Fixed.
  B: Some texts inadvertedly contained characters that caused Triton to crash. Fixed.
  N: With WB < 3.0 the reqtools screenmode requester is supported.
  C: Conversion window shows new password names instead of the old ones.
  C: DiskProtection requesters are no longer opened on the access verification screen.
  C: Added light/dark squares on both sides of the dimmer slider to make clear that "100%" means full brightness.
  C: Renamed the "DiskProtection Access Protection" to "DiskProtection Access Verification" on popular demand. (Better that way, Jussi?) :-)

- Release 1.0h, 21.09.95
  (DiskProtection V1.6, diskprot.device V1.6)

  B: At startup sometimes the access verification was unintentionally activated. Fixed.
  B: The screenblanker didn't blank the mousepointer on screens with <256 colors. Fixed.
  C: If the screen blanker is already activated when the access verification automatically pops up it this screen will be dark, too.

- Release 1.0i, 23.10.95
  (DiskProtection V1.6, diskprot.device V1.7)

  C: DPUnit conversion uses TD_PROTSTATUS only with trackdisk.device.

- Release 1.0j, 02.11.95

(DiskProtection V1.7, diskprot.device V1.7)

N: More messages in the debugging version.
B: Sometimes partitions were locked unnescessarily.
C: Changed default button in one error requester you should never see anyway ;-)

- Release 1.1, 16.11.95
  (DiskProtection V1.8, diskprot.device V1.8)

  C: Finally works with oktagon.device, too! The problem was that the task, that communicates with oktagon.device, has to be the one that opened the device. DiskProtection however implemented opening and communication in different tasks...
  N: The user can make conversion take place even if locking a filesystem fails.
  C: Reduced width of some requester texts.
  C: Memory for kicktags is allocated at the end of a memory region first. Increases stability of resident passwords.
  C: Device allows only trackdisk commands.
  B: System password was not requested when starting DiskProtection directly after reboot and this password was resident. Alas, now it is requested twice in some cases, but I prefer this solution.
- Release 1.1a, 14.01.96
  (DiskProtection V1.8, diskprot.device V1.9)

  N: a fast and unsafe encryption algorithm: SCRM (Scramble)
  B: diskprot.device has to be in DEVS: in order to be able to mount DosDrivers in the startup-sequence. Fixed in the installer script.
- Release 1.1b, 16.1.96
  (DiskProtection V1.9, diskprot.device V1.9, DPInit V0.1)

  N: DPInit
  N: ExampleXPK: example XPK sublibrary compatible with DiskProtection
  B: Sorting order in XPK sublibrary requester is no longer reversed. Why did nobody complain earlier? :-)
- Release 1.1c, 19.1.96
  (DiskProtection V1.9, diskprot.device V1.10, DPInit V0.1)

  B: SCRM was buggy and unuseable. I learned one thing: Never ever hurry up with programming at the end of an amiga meeting! ;-)
  B: Forgot to include DPInit in the archive.

- Release 1.1d, 25.05.96
  (DiskProtection V1.10, diskprot.device V1.11, DPInit V0.1)

  B: DiskProtection crashed when the locale.library became available only after diskprot.device was opened.
  B: DiskProtection takes care of [Max|Min]PkInChunk now.
  B: “Hide in File” did not find the password in short files and didn’t close the file.
  C: rewrote xpkSCRM in optimized assembler
  N: Registration Requester
- Release 1.2, 27.05.96
  (DiskProtection V1.10, diskprot.device V1.11, DPInit V1.0)

  C: bumped revisions
  C: changed DPInit icon

11 Future

There are some ideas that might find their way into future version of DiskProtection. However, right now (16.11.95) only 3 persons have registered and I have other interesting *and* profitable projects, so I'll carefully think about implementing new features...

- Make it possible to restore a DPUnit after a crash during conversion.
- Implement a fast, but because of that also unsafer algorithm.
- Hotkey(s) to remove resident passwords and/or the whole diskprotection system.
- Speed optimization
- Support more languages. I depend on your help here: My knowledge is limited to English and my native language German. You are very welcome to contact me. I will send you further information on how to create a catalog for DiskProtection then.
- If a password is not entered, opening the device should not fail, but the device should report "no disk inserted" until the password is entered. Is there a need for that?

If you have any suggestions, found some bugs or simply like the program, do not hesitate to write me. Of course I would prefer the last reason, especially if together with the message the shareware fee arrives...

12 Address of the Author

Patrick Ohly
Weechstr. 1, WG E0/1 76131 Karlsruhe
Germany

Tel.: +49 721 615662
eMail: patrick.ohly@stud.uni-karlsruhe.de
IRC: Irish

Bank Account:
Sparkasse Karlsruhe, BLZ 660 501 01
Konto–Nr. 100 621 31

Please use eMail if possible—that way you will get response more likely ;–)

13 Credits

A lot of thanks to (order without Deep Thoughts ;–):

Stefan Zeiger
: for the triton.library in general and especially for considering my wishes and suggestions

Richard Outerbridge
: for his DES source code

Angela Schmidt
: for her nice GadTools register number requester

Christian von Roques
: for xpkFEAL, informations about xpkmaster.library

André Beck
: for xpkIDEA and useful suggestions about encryption of disks

Bernhard Möllemann, Mark Rose, Samir Gajjar
: for beta testing, suggestions and criticism

Daniel Schrod
: for his efforts to convince me of the great profit of DiskProtection for mankind ;–), persistence and finally beta testing

Thomas Schröder
: for willingly lending me his Terry Pratchett books 8–)

Michael 'Mick' Hohmann
: for his NetWB–Icons (what a luck that only a few people saw my own try…)

Klaus Deppisch
: for his FFS patch

all I have forgot to mention
: for this and that

Appendix A Default Values

- Access Verification
  - Hotkey: "CTRL ALT b"
  - Period: 300 s
  - At Program Start: No
  - Screen: PAL/NTSC:LowRes, 320x200, 4 colors
  - Font: topaz 8
  - Dimmer: 100%
- System Encryption: IDEA, Mode 100
- Password Text: “” (empty string)

Appendix B Glossar

DES

Data Encryption Standard: cryptographic algorithm
standard algorithm of the U.S. government for unclassified data

DOS–device

Offers access to IO hardware on the DOS side. Organizes data in files and directories and perhaps writes them as blocks to a data medium through an exec device. Examples: '`DF0:`', '`HD0:`', '`RAM:`' (the last does not use an exec device). Are created while booting or later with '`Mount`'.

(DP)Units

Stands for a unit of the diskprot.device, but is also used for the medium encrypted with DiskProtection and the DOS–device created for this purpose. If the meaning is obvious because of the context the '`DP`' is obmitted to increase readability.

Exec Device

Offers access to IO hardware on the exec layer. Usually manages data as blocks. Examples: '`trackdisk.device`', '`ramdrive.device`'

FEAL

Fast Encryption Algorithm: a cryptographic algorithm

HASH Value

Big number created from a string, but not giving any hint on this text and highly probably different for different strings.

Medium

The actual data medium in a device. Example: floppy disk

IDEA

International Data Encryption Algorithm: a cryptographic algorithm

Volume

A single data medium as used from DOS. Is accessed by its name, no matter if it is inserted in any device right now. Examples: A certain formated floppy disk, '`Workbench:`' in contrast to '`HD0:`'.

Appendix C Index

Table of Contents