# Eraser Help Index

# Legal



By using this software you accept all the terms and conditions of the license agreement and disclaimer below.

**License agreement**

Eraser is a freeware program and must be provided at no charge. Feel free to share this program with your friends, but please do not give it away altered or as part of another system.

**Disclaimer**

Eraser is provided on an "AS IS" basis, without warranty of any kind either express or implied, including warranties of merchantability or fitness for a particular purpose. In no event will the author be liable to you for damages, direct or consequential, which may result from the use of this program. The user must assume the entire risk of using the software.

## Files

### Files included in archive

| | |
|---|---|
| file_id.diz | program information for distributors |
| readme.txt | program information for the user |
| setup.exe | the setup program |

### Files installed by the setup program

| | |
|---|---|
| eraser.exe | normal user interface |
| erasext.dll | shell extension |
| scheduler.exe | scheduler |
| eraser.dll | file erasing library (System directory) |
| eraser.hlp | help file |
| eraser.cnt | help contents file |
| stuninstall.exe | uninstall program (System directory) |
| uninstall.dat | uninstall information |

## Introduction

Eraser consists of several utilities built on top of a powerful file wiping library capable of securely removing sensitive data from your hard drive.

Three user interfaces are included – a standard wipe utility, a convenient Explorer shell extension and a scheduler which allows you to program the erasing of unused disk space or, for example, cache files to happen regularly, at night, during your lunch break, at weekends or whenever you like.

By default files are overwritten several times with carefully selected patterns, based on Peter Gutmann's paper Secure Deletion of Data from Magnetic and Solid-State Memory, which effectively remove the magnetic remnants from the hard disk making it impossible to recover the data.

For faster (and less effective) file erasing, Eraser also supports a method defined in the National Industrial Security Program Operating Manual of the US Department of Defense.

There is also an option to overwrite only with pseudo-random data.

---

**See also:**
**Disclaimer**

## Setup

### Installation

Run `setup.exe` to install Eraser. The setup program will extract the files, copy them to a desired location and create shortcuts to the Start menu (if you like).

Eraser requires `mfc42.dll` and `msvcrt.dll`, which can be obtained at Eraser home page.

### Removal

To uninstall Eraser go to Add / Remove Programs in the Control Panel. You should not delete files manually, if you do, created registry entries will not be removed.

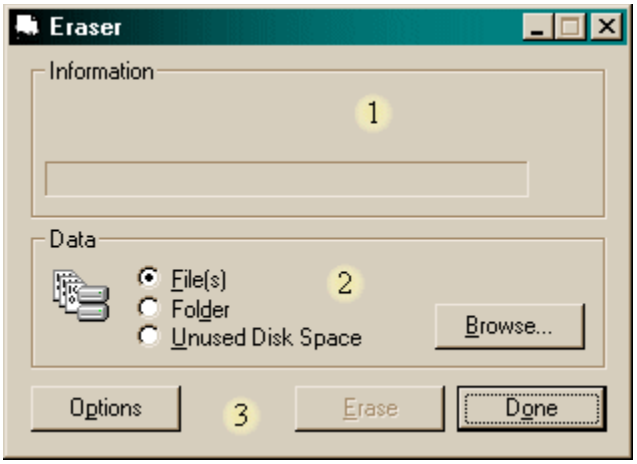
# User Interface: Normal

The normal user interface allows you to run Eraser as a separate program. You can start it by running `eraser.exe`.

---



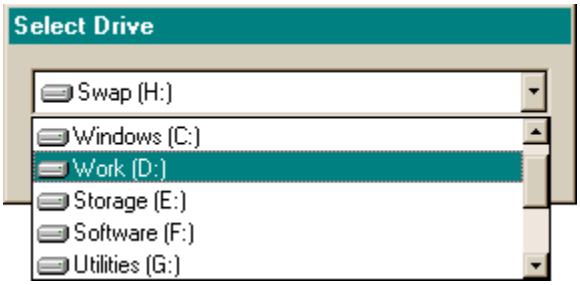
### User Interface




1. **Information section**
   – Shows you information about the current process (see below).
2. **Data section**
   – Allows you to select files, a folder or a hard drive to be erased.
3. **Control**
   – Allows you to change the erasing method and to start or stop the erasing process.

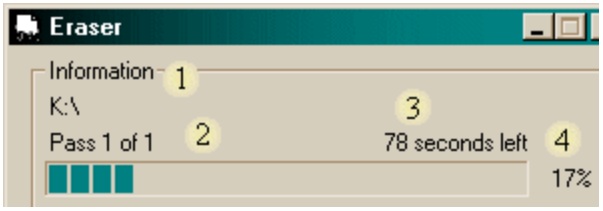
### Erasing Data

You may choose one or several files, a folder or unused space on a drive to be erased by selecting the appropriate data type on the data section and clicking Browse.



After you have selected the data to be erased, you may start the process by clicking Erase. The previously selected data will be reset every time you change the data settings. Please notice that if you select to erase a folder, all data in that folder and its subfolders will be deleted.

The information section provides you information about the current file:

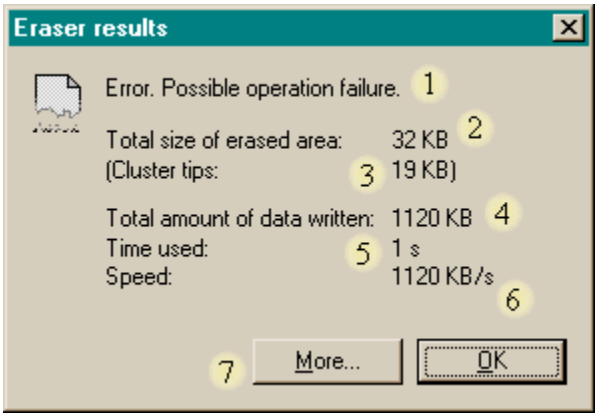



1. **File name**
   – The file currently being erased (if you are erasing unused disk space, this will show you the drive letter)
2. **Pass**
   – Shows you how many passes there will be and which one is being written.
   – When you are erasing unused disk space, Eraser will create several smaller files instead of one huge file to improve the performance. Therefore this value shows you the pass which is being written to the current temporary file.
3. **Time**
   – Shows you how long the erasing process will approximately take or possibly another information about the current process.
4. **Progress**
   – Shows you how much of the current file has been processed.

You can stop the erasing process any time by clicking Stop.

**After Erasing**

After the operation, results will be shown in a dialog:




1. **Information**
   – Tells you if the operation was successful. Possible values:

`Finished successfully.`
– The operation was successful. The selected data was wiped and deleted.

`Process terminated.`
– The user terminated the operation.

`Error. Possible operation failure.`
– Eraser was unable to wipe and delete one or more of the selected files or unused space on one or more of the selected items. You can get more information by clicking More.

2. **Total size of erased area**
   – The size of the area erased (in kilobytes – including cluster tip area).
3. **Cluster tips**
   – The size of the unused area allocated by files. Shows you how much of possibly secret data may have been on your disk without you ever knowing.

---

**See also:**
**Erasing: Cluster tips**

4. **Total amount of data written**
   – Shows you the amount of data that was written to the disk (in kilobytes).
5. **Time used**
   – Shows you how long writing the data to the disk took (in seconds).
6. **Speed**
   – Shows you the average write speed (kilobytes / second)
7. **More**
   – If errors occurred, you can get more information by clicking this button.

   – The following dialog will appear showing you a list of items which Eraser was unable to properly process.

1. **List of items (files)**

**Running from Command Line**

You can also launch normal user interface from the command line. The usage:

```
eraser [-file | -folder | -disk data] [-silent] [-options]

-file        data to erase is a file (wildcards may be used)
-folder      data to erase is a folder (including subfolders)
-disk        data to erase is unused space on a drive
-silent      no windows will be shown
-options     ignores all other (valid) parameters and shows options dialog
```

Examples:

Erase all files in the directory C:\Windows\Temp leaving subfolders untouched:

```
eraser -file C:\Windows\Temp\*
```

Erase folder H:\Some Folder and all its files and subfolders:

```
eraser -folder "H:\Some Folder"
```

Erase unused disk space on drive L without showing any windows:

```
eraser -disk L:\ -silent
```

Show Eraser Options dialog:

```
eraser -options
```

---

**See also:**

**User interface: shell extension**
**User interface: scheduler**

# User Interface: Shell Extension

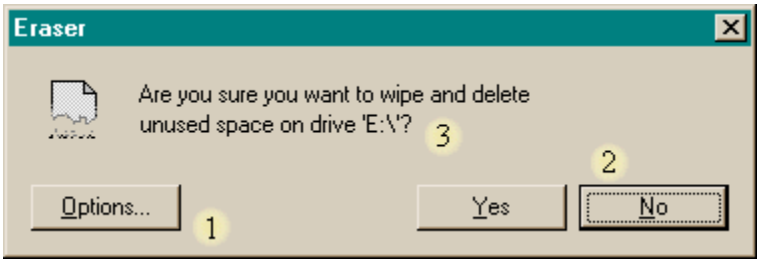
Shell extension allows you to run Eraser from Windows Explorer. You can erase files and folders or unused space on one or more drives by right clicking them and choosing Erase from the appearing context menu.

---



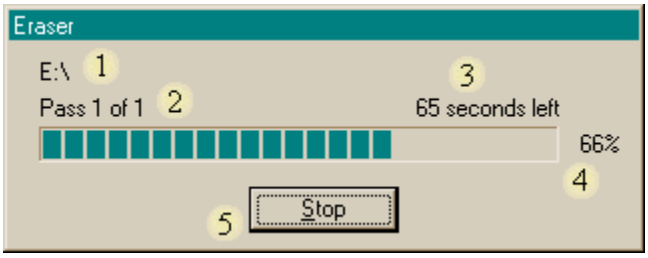
### Erasing Data

Eraser will ask you to confirm the procedure before starting:




1. **Options**
   – Allows you to change the erasing method.
2. **Confirmation**
   – Start erasing by clicking Yes; cancel the process by clicking No. If you have selected to erase unused space on multiple drives, Yes to All will also be available.
3. **Data**
   – Shows you the data you are about to erase.

After clicking Yes, Eraser will show you an information dialog during the whole operation:



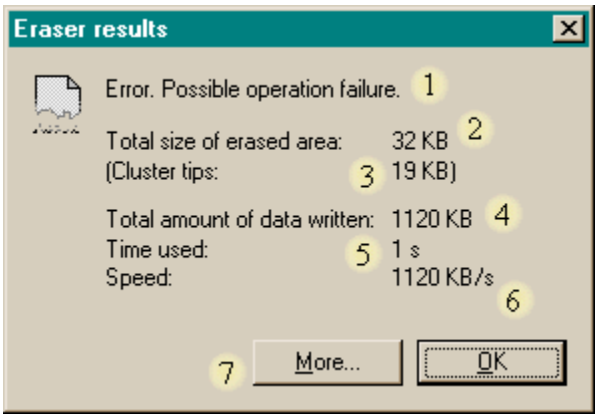

1. **File name**
   – The file currently being erased (if you are erasing unused disk space, this will show you the drive letter)
2. **Pass**
   – Shows how many passes there will be and which one is being written.
   – When you are erasing unused disk space, Eraser will create several smaller files instead of one huge file to improve the performance. Therefore this value shows you the pass which is being written to the current temporary file.

3. **Time**
   – Shows you how long the erasing process will approximately take or possibly another information about the current process.
4. **Progress**
   – Shows you how much of the current file has been processed.
5. **Stop**
   – Interrupts the erasing process.

You can stop the erasing process any time by clicking Stop.

**After Erasing**

After the operation, results will be shown in a dialog:




1. **Information**
   – Tells you if the operation was successful. Possible values:

   `Finished successfully.`
   – The operation was successful. The selected data was wiped and deleted.

   `Process terminated.`
   – The user terminated the operation.

   `Error. Possible operation failure.`
   – Eraser was unable to wipe and delete one or more of the selected files or unused space on one or more of the selected items. You can get more information by clicking More.

2. **Total size of erased area**
   – The size of the area erased (in kilobytes – including cluster tip area).
3. **Cluster tips**
   – The size of the unused area allocated by files. Shows you how much of possibly secret data may have been on your disk without you ever knowing.

---

**See also:**

4. **Total amount of data written**
   – Shows you the amount of data that was written to the disk (in kilobytes).
5. **Time used**
   – Shows you how long writing the data to the disk took (in seconds).
6. **Speed**
   – Shows you the average write speed (kilobytes / second)
7. **More**
   – If errors occurred, you can get more information by clicking this button.

   – The following dialog will appear showing you a list of items which Eraser was unable to properly process.


**1. List of items (files)**

**Note!** The results will not be shown if you have selected to erase unused disk space on multiple drives and have clicked Yes to All when Shell Extension was asking for your confirmation.

---

**See also:**
**User interface: normal**
**User interface: scheduler**

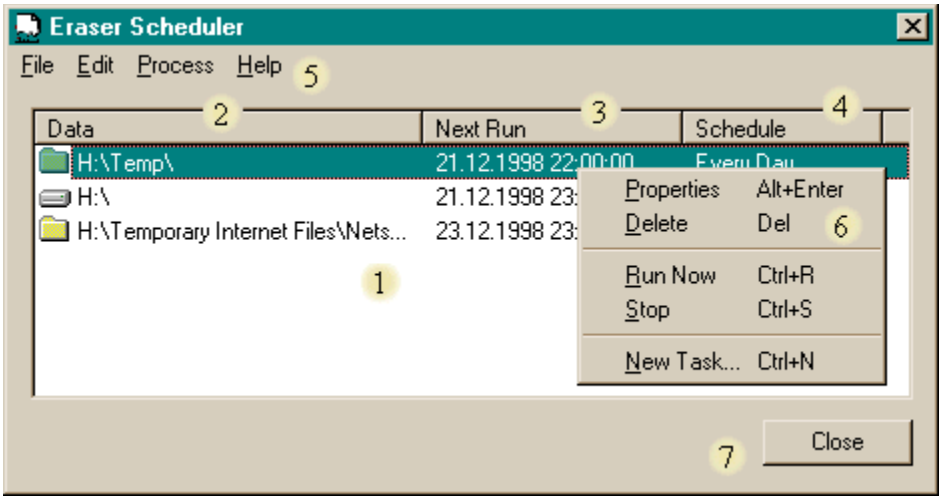
# User Interface: Scheduler

Eraser Scheduler allows you to schedule the erasing of unused disk space or e.g. cache files to happen regularly, at night, during your lunch break, at weekends or whenever you like. You can start Scheduler by running `scheduler.exe`.

---



## User Interface




1. **Assignment List**
   – List of scheduled assignments
2. **Data**
   – This column tells you the data which will be erased, i.e. a drive or a folder.
3. **Next Run**
   – Shows you when the assignment is scheduled to run next. When Scheduler is running the assignment, text "Running Now…" will be shown.
4. **Schedule**
   – Shows you when you have scheduled the assignment to be processed, for example "Every Day".
5. **Menu**
   **File**
   – **New Task** (press **Ctrl+N**), creates a new assignment
   – **View Log** (press **Ctrl+L**), executes an application to view the log file.
   – **Import**, imports tasks from a file

– **Export**, exports scheduled tasks into a file
– **Close** (press **Alt+F4**), closes the Scheduler window but does not quit the program
– **Exit**, quits the program

**Edit**

– **Properties** (press **Alt+Enter**), open the property window for the selected assignment.
– **Delete** (press **Del**), removes the selected assignment from the list
– **Preferences / Scheduler** (press **Ctrl+P**), allows you to edit Scheduler preferences
– **Preferences / Eraser Options** (press **Ctrl+E**), allows you to edit Eraser preferences
– **Update List** (press **F5**), refreshes the assignment list

**Process**

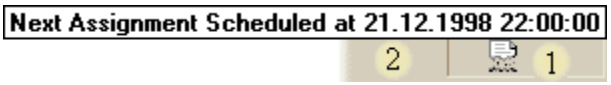
– **Run Now** (press **Ctrl+R**), executes the selected assignment immediately
– **Stop** (press **Ctrl+S**), interrupts a running assignment

**Help**

– **Help**, launches the help file
– **About Scheduler**, shows program information

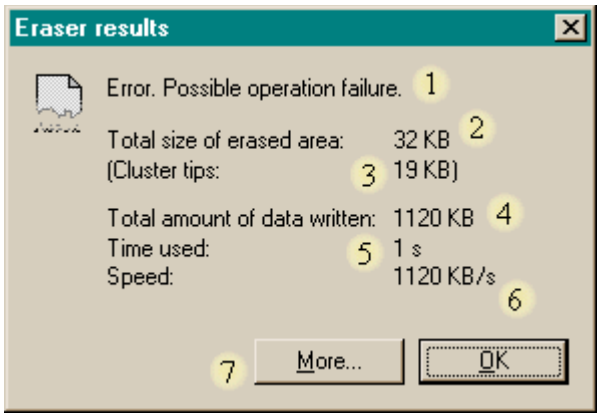
6. **Pop-up Menu**
   – Appears when you right click the assignment list.
   – Contains the most important menu items allowing you to quickly edit the selected assignment.
   – See explanations of the menu items above.
7. **Close**
   – Closes the Scheduler window but does not exit the program.

Scheduler starts as a taskbar tray application, you can open the main window by double-clicking the tray icon added to the task bar tray area.




1. **Icon**
   – Right click to show the menu, double-click to open the Scheduler window
   – The icon changes depending on the current state of the Scheduler:

   – The icon above means that Scheduler is ready and waiting for an assignment.




**1.** Scheduler is running one or more assignments.


**1.** Scheduler is disabled (i.e. it ignores scheduled assignments).

**2. Tool Tip**

– Shows you information about the current state of the Scheduler.

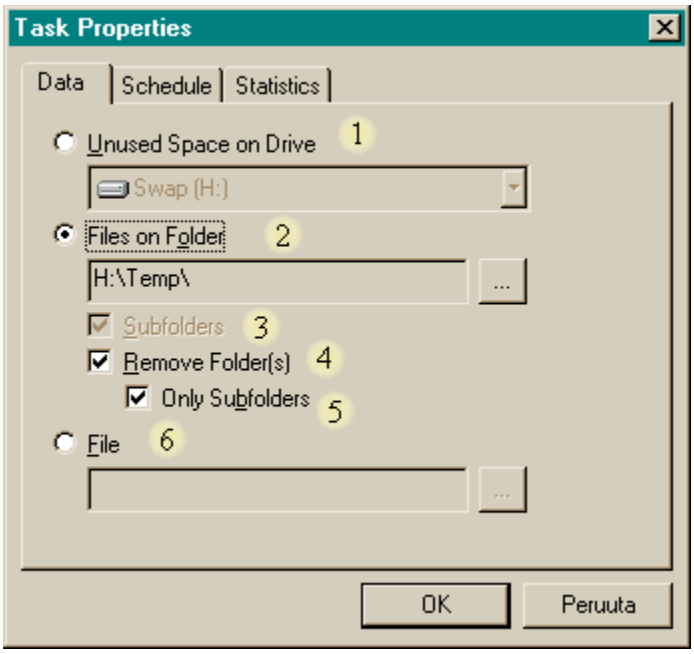

**1. Tray Icon Menu**

– Right click the icon to show the menu

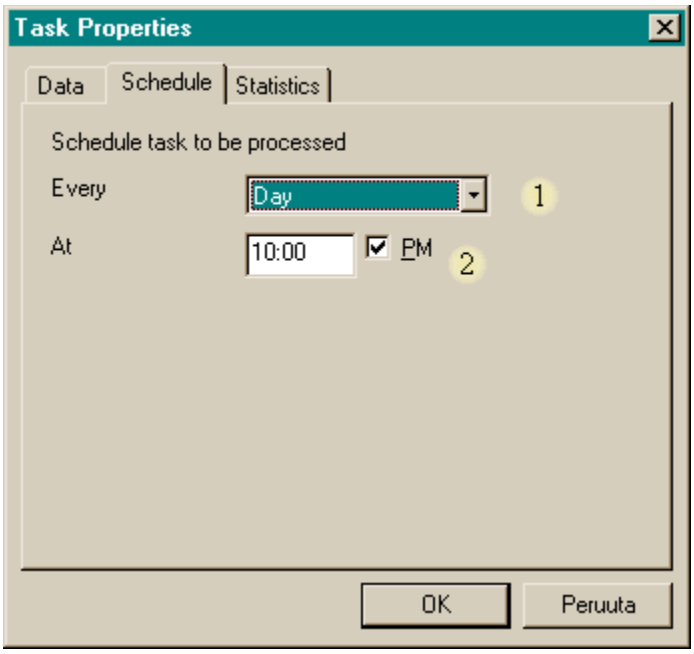
– **Open..** (double click), opens the Scheduler window.
– **Enabled**, allows you to change the Scheduler state from enabled to disabled and vice versa (disabled means that Scheduler ignores scheduled assignments).
– **Exit**, quits Scheduler.

**Creating Assignments**

To create an assignment, select New Task from the menu, or press Ctrl+N. A dialog appears allowing you to edit the properties of the newly created assignment.

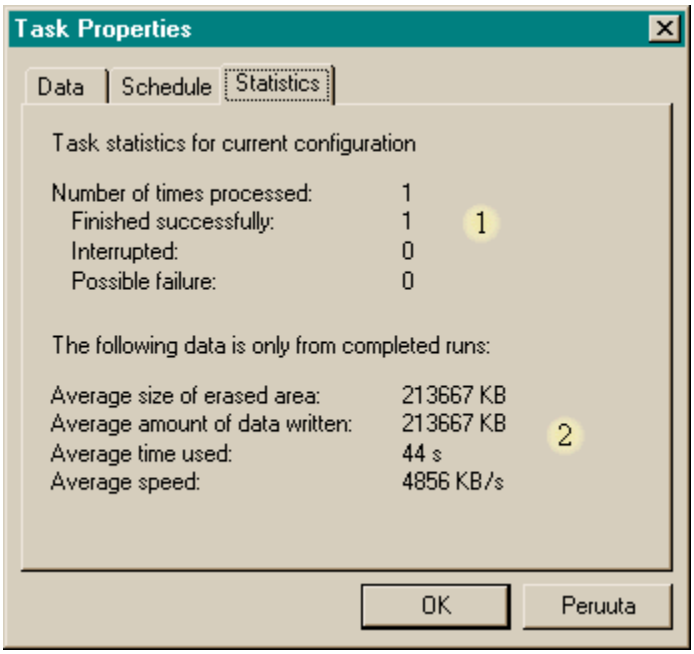



1. **Unused Space on Drive**
   –If you want to erase unused disk space on a drive, select this option and choose a drive from the list.
2. **Files on Folder**
   – If you instead want to erase all files on a folder, select this option and press the button to browse to a desired directory.
3. **Subfolders**
   – If you want to erase files only from the selected folder and leave subfolders untouched, you must deselect this option.
4. **Remove Folder**
   – If you have selected to erase files on a folder, the folder will not be removed unless you select this option.
5. **Only Subfolders**
   – If you also select this option, only subfolders will be removed (if there are any).
6. **File**
   – By selecting this option you may click the button to browse to a file to be erased. Useful when overwriting, for example, web browser's history file or news reader database.




1. **Every…**
   – Select whether you wish to perform the assignment daily or weekly (and which day of the week).
2. **At…**
   – Select at what time of the day you would like the assignment to be processed. The check box allowing you to choose from AM and PM is available only if Windows is set to have 12-hour clock.

The Statistics page can later be used to view task statistics, you can ignore it when creating new assignments.




1. **Counters**

– Shows you how many times the selected assignment has been processed and what were the results.

2. **Statistics**
   – Various statistical figures for the selected assignment.
   – Statistics will be reset every time you change the data to be erased.

---

**See also:**
**User Interface: Normal: After Erasing**
**User Interface: Shell Extension: After Erasing**

After choosing the desired data to be erased and setting schedule, you can save the assignment by clicking OK.

### Editing Assignments and Viewing Statistics

Select an assignment from the list and press Alt+Enter, or select Properties from the menu. Scheduler will show you the same dialog as it did when you created the assignment allowing you to edit all assignment properties.

You can view task statistics by opening the Statistics page from the Properties window. Statistics will be reset every time you change the data to be erased.

### Removing Assignments

Select an assignment from the list and press Del, or select Delete from the menu.

### Running and Interrupting Assignments

If you do not want to wait for an assignment to be processed, you can select an assignment from the list and press Ctrl+R, or select Run Now from the menu.

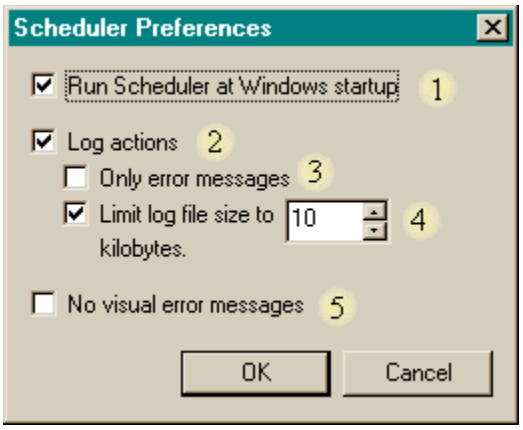
To interrupt an assignment, scheduled or not, select it from the list and press Ctrl+S, or select Stop from the menu.

### Enabling and Disabling Scheduler

If you do not want Scheduler to process assignments for now, right click the tray icon and deselect the Enabled option. The tray icon and the tool tip will change to signal that Scheduler is disabled.

### Editing Scheduler Preferences

To edit Scheduler preferences, press Ctrl+P, or select Preferences / Scheduler from the menu. The following dialog will appear.




1. **Run Scheduler at Windows startup**
   – Select this if you want Scheduler to run automatically every time you start Windows.
2. **Log Actions**
   – If you select this, Scheduler will write all important actions to a log file.
3. **Only Error Messages**
   – Select this if you want only error messages that would prevent an assignment to be executed properly to be logged.
4. **Limit Log File Size**
   – Select this if you want to limit the size of the log file.
5. **No Visual Errors**
   – If you select this, Scheduler will not prompt you of errors that happened while processing an assignment. You will be prompted of errors which may occur while e.g. creating a new task though.
   – You should select this if Scheduler is running a long time (say overnight or longer) without anyone watching and dismissing error messages.
   – If you select this, you should also select to log actions (or at least error messages).

By default, all of these options except limit log file size are selected.

### Using the Log Actions Option

If you have selected to Log Actions, Scheduler will write a log file called `schedlog.txt` to the same directory where the executable is located.

Here is a sample of a log file.

```
21.12.1998 08:30:49: Scheduler starting.
21.12.1998 11:30:00: Running assignment – H:\Temp\.
21.12.1998 11:33:19: Assignment finished - H:\Temp\.
21.12.1998 17:00:00: Running assignment – H:\.
21.12.1998 17:01:42: Possible operation failure running assignment - H:\.
21.12.1998 17:01:42: Failed to wipe unused space on H:\WIN386.SWP.
22.12.1998 11:30:00: Running assignment – H:\Temp\.
22.12.1998 11:30:13: Assignment finished - H:\Temp\.
22.12.1998 17:00:00: Running assignment – H:\.
```

```
22.12.1998 17:02:14: Possible operation failure running assignment - H:\.
22.12.1998 17:02:14: Failed to wipe unused space on H:\WIN386.SWP.
22.12.1998 17:47:59: New assignment created - E:\.
22.12.1998 17:48:15: New assignment created - D:\Project Files\.
22.12.1998 23:00:00: Running assignment – D:\Project Files\.
22.12.1998 23:45:00: Running assignment – E:\.
22.12.1998 23:46:59: Assignment finished - D:\Project Files\.
22.12.1998 23:58:28: Assignment finished - E:\.
23.12.1998 11:30:00: Running assignment – H:\Temp\.
23.12.1998 11:31:57: Assignment finished - H:\Temp\.
23.12.1998 15:18:27: Removed an assignment - D:\Project Files\.
23.12.1998 17:00:00: Running assignment – H:\.
23.12.1998 17:06:38: Possible operation failure running assignment - H:\.
23.12.1998 17:06:38: Failed to wipe unused space on H:\WIN386.SWP.
23.12.1998 20:28:22: Scheduler quitting.
```

If you have selected to log only errors, only errors that would prevent an assignment to be executed properly (red) would have been written to the log file.

---

**See also:**
**User interface: normal**
**User interface: shell extension**

## Erasing: How Is It Done

You can choose from three overwriting methods.

The default method is the most efficient and should be used to achieve maximum security. It is recommend to use this option if there is a possibility that someone will open the hard drive and examine it with sophisticated equipment.

If you only want to keep curious hackers or other users from reading your files, it is sufficient to use either of the remaining options. Even overwriting once with pseudo-random data will make the data impossible to recover with any disk utility.

When you start the wiping process, Eraser first fills the write buffer with suitable data (depending on the selected method). The buffer used is larger than in most wipe utilities to improve the performance.

Files can be overwritten up to 35 times. After each write, all buffers are flushed so data will be written directly to the disk instead of being cached by the operating system. If overwriting was performed successfully, the file will be truncated to zero length and its date will be changed before deleting. If overwriting was not successful, the file will not be deleted.

**Note!** Even though Eraser wipes the body of a file so that it cannot be recovered, it does not wipe the file name (from the file system table) **when running Windows 95/98**. This may be a problem if you do not want anyone to know that the file was on the disk. To remove the file names after erasing, run Defrag.

**Warning!** After your confirmation, Eraser wipes and deletes files even if they are marked as read-only.

---

**See also:**
**Erasing: Options**
**Overwriting properly**
**Government regulations**

# Erasing: Files and Folders

### Using the normal user interface

1. Select Files from the data section. (If you want to erase folders, use the shell extension)
2. Click Browse and select the files you wish to erase.
3. Click Erase to start the process.
4. After the operation, the results will be shown in a dialog. If errors occurred, you can get more information by clicking More.

### Using the shell extension

1. Using the Explorer, select the files and/or folders you wish to erase. Be careful when erasing folders, all subfolders will be erased too.
2. Right click while the mouse pointer is over the selection.
3. Select Erase from the context menu.
4. Click Yes to start the process.
5. Ensure that all files and folders were erased. If a file could not be erased, it was not deleted.

### Scheduling file erasing

1. Select New Task from the menu.
2. Select Files on Folder and click the button to browse to a desired folder or if you want erase only one file, select File and browse to the desired file.
3. If you want to remove the folder, select Remove Folder.
4. Schedule when you want to run the assignment (daily or weekly and at what time).
5. Click OK to save the assignment.

---

**See also:**
**User interface: normal**
**User interface: shell extension**
**User interface: scheduler**

# Erasing: Unused Disk Space

### Using the normal user interface

1. Select Unused Disk Space from the data section.
2. Click Browse and select the drive whose unused space you wish to erase.
3. Click Erase to start the process.
4. After the operation, the results will be shown in a dialog. If errors occurred, you can get more information by clicking More.

### Using the shell extension

1. Using the Explorer, select the drive(s) whose unused space you wish to erase.
2. Right click while the mouse pointer is over the selection.
3. Select Erase unused space from the context menu.
4. Click Yes to start the process (If you have selected multiple drives, you can also click Yes to All. Otherwise Eraser will ask your confirmation for every selected drive).

### Scheduling erasing of unused disk space

1. Select New Task from the menu.
2. Select Unused Space on Drive and select a drive from the list.
3. Schedule when you want to run the assignment (daily or weekly and at what time).
4. Click OK to save the assignment.

---

**See also:**
**User interface: normal**
**User interface: shell extension**
**User interface: scheduler**
**Erasing: Cluster tips**

## Erasing: Cluster Tips

To be able to keep record of the drive contents, the file system divides the disk to small blocks called clusters. A cluster is the smallest data block, which can be allocated from the disk. The size and the amount of clusters on a partition depend on the file system and the size of the partition.

It is relatively rare for the size of a file to be divisible by the cluster size, i.e. for the file to completely use all clusters it has allocated. Therefore, usually only a part of the last allocated cluster is used. The unused part of the last cluster contains old and possibly secret data, which cannot be overwritten before the file that allocated the cluster is removed.

If you select to Erase cluster tips, Eraser will overwrite the unused space at the end of the last cluster of each file on the selected drive before overwriting the unused disk space.

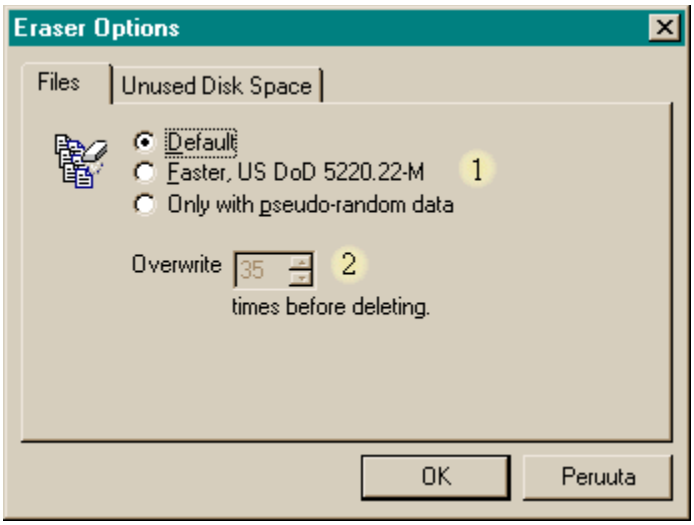
**Note!** Cluster tips of files that are locked (in use) cannot be overwritten and you will receive an error because of this. To unlock as many files as possible, close all running applications before erasing unused disk space.

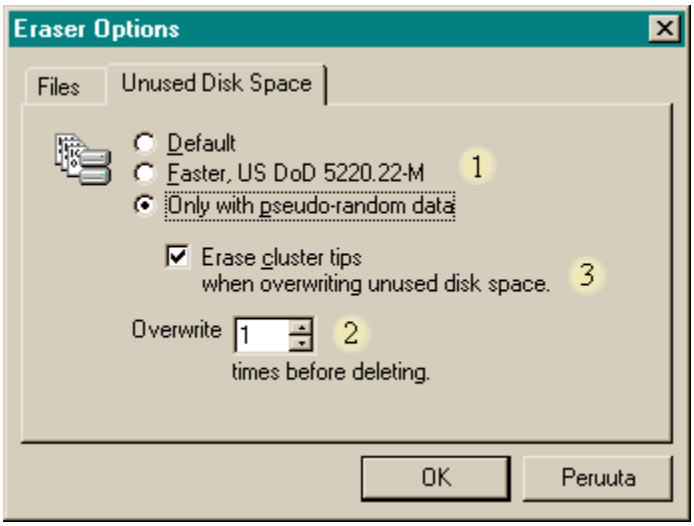
---

**See also:**
**Erasing: Unused disk space**
**Erasing: Options**

## Erasing: Options

Eraser allows you to use separate settings for overwriting files and unused disk space.




1. **Method**
   Three methods for overwriting are available:

   Default
   – 35 pass method based on Secure Deletion of Data from Magnetic and Solid-State Memory. For ultimate security.

   Faster
   – 7-pass method based on National Industrial Security Operating Manual by U.S. Department of Defense.

   Only with pseudo-random data
   – 1 to 35 passes of data created with additive congruential prng. This method is default for erasing unused disk space.

---

**See also:**
**Advanced section starting at Deleting files**
**Erasing: How is it done**

2. **Passes**
   – If you have selected to erase only with pseudo-random data, you may choose how many times to overwrite before deleting
3. **Cluster tips**
   – If you select this, Eraser will wipe the unused area at the end of the last cluster of every file on a drive when erasing unused disk space.

---

**See also:**
**Erasing: Cluster tips**

**Note!** Once you change the settings, they will be used no matter which user interface you use.

## Deleting Files

An operating system, such as MS-DOS and Windows, uses a file system to keep track of the data on the hard disk: directories, files, their location, size, dates etc. Using a file system is noticeably faster than parsing the information directly from the disk, but also causes problems when the user wants to securely remove a file.

Normally, when you delete a file, the operating system does not actually erase the file; it only removes the reference of the file from the file system table and marks the area unused. Therefore anyone can recover the file using any disk maintenance utility capable of reading the disk directly. The data will not be destroyed until another program writes over the deleted file, and even after that it may be possible to recover some or all of the data by studying the disk with specialized equipment.

To most people this is enough, but if you want to ensure that your confidential data will not end up in wrong hands, you should properly overwrite a file before removing its reference from the file system table.

---

**See also:**
**Overwriting properly**

## Overwriting Properly

A hard disk consists of one or several disk platters which have been plated with a very thin (a few millionths of an inch thick) layer of magnetic substance. One read/write head is being used to both read and write data from the platter. The head is positioned very close to the platter, only 3-7 millionths of an inch away. The surface of the disk platter can be seen to consist of magnetic domains acting like small magnets, having both positive and negative poles. The data is saved to the disk in binary form - as ones and zeros - and millions of magnetic domains are used to save one bit. When writing new data to the disk, the read/write head reverses the magnetic pole direction if necessary.

When the read/write head reverses the polarity of a region of domains (presenting one bit of data), the polarity of most domains reverse, but a small portion remain in their original state. The electronics of the drive ignore these small inaccuracies, but when studying the platter surface with a sophisticated microscope it may be possible to recover data even if it has been overwritten.

The main purpose of overwriting is to alter the magnetic polarity of each domain on the disk platter as much as possible so it will be extremely hard to determine their previous state.

If the data was written directly to the disk, files could simply be overwritten with patterns consisting only of ones or zeros. However, various run-length limited encoding algorithms are used in hard drives to prevent read/write head from losing its position. This causes that only limited amount of adjacent ones or zeros will be written to the disk. That is why different encoding schemes must be taken into account when selecting overwriting patterns.

In his paper Secure Deletion of Data from Magnetic and Solid-State Memory, Peter Gutmann has discussed the subject further. In chapter Erasure of Data stored on Magnetic Media he suggests a 35 pass overwriting process which should erase the data despite the drive encoding. This is the process Eraser uses as its default overwriting method.

The first four and last four passes are pseudo-random data (created with additive congruential pseudo-random number generator) and other passes are made in random order. This overwriting method is recommend for maximum security.

---

**References:**

**Quantum Storage Resources**

**IBM Storage**

**Peter Gutmann, Secure Deletion of Data from Magnetic and Solid-State Memory**

**Donald Ervin Knuth, The Art of Computer Programming: Seminumerical Algorithms, Volume 2, 3rd Edition**

## Government Regulations (for Secure Data Removal)

Governments often have need to destroy classified information, so several regulations concerning the subject have been made. To mislead opponents, publicly available regulations may intentionally underrate the destruction methods, while the real regulations remain classified. Many of the available regulations are also quite old.

Eraser offers an overwriting method defined in the National Industrial Security Program Operating Manual (**NISPOM a.k.a. DoD 5220.22-M**) of the US Department of Defense (**January 1995; chapter 8, section 3, 8-306. Maintenance**).

The data will be overwritten seven times to all addressable locations:

| Pass | Data |
|---|---|
| **1** | A random character, n = [0, 255] |
| **2** | A random character, n |
| **3** | Complement of previous character, ~n |
| **4** | A random character, n |
| **5** | A random character, n |
| **6** | Complement of previous character, ~n |
| **7** | A random character, n |

**Note!** It is not recommend using this overwriting method if there is a reason to believe that the hard drive will be examined with specialized equipment. For normal overwriting, this method is quite adequate.

**Note!** This method is not suitable for overwriting files or unused disk space on a compressed drive.

---

**References:**

[National Industrial Security Program Operating Manual (NISPOM)]

# Secure Deletion of Data from Magnetic and Solid-State Memory

Peter Gutmann
*Department of Computer Science*
*University of Auckland*
pgut001@cs.auckland.ac.nz

This paper was first published in the Sixth USENIX Security Symposium Proceedings, San Jose, California, July 22-25, 1996

## Abstract

With the use of increasingly sophisticated encryption systems, an attacker wishing to gain access to sensitive data is forced to look elsewhere for information. One avenue of attack is the recovery of supposedly erased data from magnetic media or random-access memory. This paper covers some of the methods available to recover erased data and presents schemes to make this recovery significantly more difficult.

Secure Deletion of Data from Magnetic and Solid-State Memory

# Introduction

Much research has gone into the design of highly secure encryption systems intended to protect sensitive information. However work on methods of securing (or at least safely deleting) the original plaintext form of the encrypted data against sophisticated new analysis techniques seems difficult to find. In the 1980's some work was done on the recovery of erased data from magnetic media [1] [2] [3], but to date the main source of information is government standards covering the destruction of data. There are two main problems with these official guidelines for sanitizing media. The first is that they are often somewhat old and may predate newer techniques for both recording data on the media and for recovering the recorded data. For example most of the current guidelines on sanitizing magnetic media predate the early-90's jump in recording densities, the adoption of sophisticated channel coding techniques such as PRML, the use of magnetic force microscopy for the analysis of magnetic media, and recent studies of certain properties of magnetic media recording such as the behaviour of erase bands. The second problem with official data destruction standards is that the information in them may be partially inaccurate in an attempt to fool opposing intelligence agencies (which is probably why a great many guidelines on sanitizing media are classified). By deliberately under-stating the requirements for media sanitization in publicly-available guides, intelligence agencies can preserve their information-gathering capabilities while at the same time protecting their own data using classified techniques.

This paper represents an attempt to analyse the problems inherent in trying to erase data from magnetic disk media and random-access memory without access to specialised equipment, and suggests methods for ensuring that the recovery of data from these media can be made as difficult as possible for an attacker.

# Methods of Recovery for Data stored on Magnetic Media

Magnetic force microscopy (MFM) is a recent technique for imaging magnetization patterns with high resolution and minimal sample preparation. The technique is derived from scanning probe microscopy (SPM) and uses a sharp magnetic tip attached to a flexible cantilever placed close to the surface to be analysed, where it interacts with the stray field emanating from the sample. An image of the field at the surface is formed by moving the tip across the surface and measuring the force (or force gradient) as a function of position. The strength of the interaction is measured by monitoring the position of the cantilever using an optical interferometer or tunnelling sensor.

Magnetic force scanning tunneling microscopy (STM) is a more recent variant of this technique which uses a probe tip typically made by plating pure nickel onto a prepatterned surface, peeling the resulting thin film from the substrate it was plated onto and plating it with a thin layer of gold to minimise corrosion, and mounting it in a probe where it is placed at some small bias potential (typically a few tenths of a nanoamp at a few volts DC) so that electrons from the surface under test can tunnel across the gap to the probe tip (or vice versa). The probe is scanned across the surface to be analysed as a feedback system continuously adjusts the vertical position to maintain a constant current. The image is then generated in the same way as for MFM [4] [5]. Other techniques which have been used in the past to analyse magnetic media are the use of ferrofluid in combination with optical microscopes (which, with gigabit/square inch recording density is no longer feasible as the magnetic features are smaller than the wavelength of visible light) and a number of exotic techniques which require significant sample preparation and expensive equipment. In comparison, MFM can be performed through the protective overcoat applied to magnetic media, requires little or no sample preparation, and can produce results in a very short time.

Even for a relatively inexperienced user the time to start getting images of the data on a drive platter is about 5 minutes. To start getting useful images of a particular track requires more than a passing knowledge of disk formats, but these are well-documented, and once the correct location on the platter is found a single image would take approximately 2-10 minutes depending on the skill of the operator and the resolution required. With one of the more expensive MFM's it is possible to automate a collection sequence and theoretically possible to collect an image of the entire disk by changing the MFM controller software.

There are, from manufacturers sales figures, several thousand SPM's in use in the field today, some of which have special features for analysing disk drive platters, such as the vacuum chucks for standard disk drive platters along with specialised modes of operation for magnetic media analysis. These SPM's can be used with sophisticated programmable controllers and analysis software to allow automation of the data recovery process. If commercially-available SPM's are considered too expensive, it is possible to build a reasonably capable SPM for about US$1400, using a PC as a controller [6].

Faced with techniques such as MFM, truly deleting data from magnetic media is very difficult. The problem lies in the fact that when data is written to the medium, the write head sets the polarity of most, but not all, of the magnetic domains. This is partially due to the inability of the writing device to write in exactly the same location each time, and partially due to the variations in media sensitivity and field strength over time and among devices.

In conventional terms, when a one is written to disk the media records a one, and when a zero is written the media records a zero. However the actual effect is closer to obtaining a 0.95 when a zero is overwritten with a one, and a 1.05 when a one is overwritten with a one. Normal disk circuitry is set up so that both these values are read as ones, but using specialised circuitry it is possible to work out what previous "layers" contained. The recovery of at least one or two layers of overwritten data isn't too hard to perform by reading the signal from the analog head electronics with a high-quality digital sampling oscilloscope, downloading the sampled waveform to a PC, and analysing it in software to recover the previously recorded signal. What the software does is generate an "ideal" read signal and subtract it from what was actually read, leaving as the difference the remnant of the previous signal. Since the analog circuitry in a commercial hard drive is nowhere near the quality of the circuitry in the oscilloscope used to sample the signal, the ability exists to recover a lot of extra information which isn't exploited by the hard drive electronics (although with newer channel coding techniques such as PRML (explained further on) which require extensive amounts of signal processing, the use of simple tools such as an oscilloscope to directly recover the data is no longer possible).

Using MFM, we can go even further than this. During normal readback, a conventional head averages the signal over the track, and any remnant magnetization at the track edges simply contributes a small percentage of noise to the total signal. The sampling region is too broad to distinctly detect the remnant magnetization at the track edges, so that the overwritten data which is still present beside the new data cannot be recovered without the use of specialised techniques such as MFM or STM (in fact one of the "official" uses of MFM or STM is to evaluate the effectiveness of disk drive servo-positioning mechanisms) [7]. Most drives are capable of microstepping the heads for internal diagnostic and error recovery purposes (typical error recovery strategies consist of rereading tracks with slightly changed data threshold and window offsets and varying the head positioning by a few percent to either side of the track), but writing to the media while the head is off-track in order to erase the remnant signal carries too much risk of making neighbouring tracks unreadable to be useful (for this reason the microstepping capability is made very difficult to access by external means).

These specialised techniques also allow data to be recovered from magnetic media long after the read/write head of the drive is incapable of reading anything useful. For example one experiment in AC erasure involved driving the write head with a 40 MHz square wave with an initial current of 12 mA which was dropped in 2 mA steps to a final level of 2 mA in successive passes, an order of magnitude more than the usual write current which ranges from high microamps to low milliamps. Any remnant bit patterns left by this erasing process were far too faint to be detected by the read head, but could still be observed using MFM [8].

Even with a DC erasure process, traces of the previously recorded signal may persist until the applied DC field is several times the media coercivity [9].

Deviations in the position of the drive head from the original track may leave significant portions of the previous data along the track edge relatively untouched. Newly written data, present as wide alternating light and dark bands in MFM and STM images, are often superimposed over previously recorded data which persists at the track edges. Regions where the old and new data coincide create continuous magnetization between the two. However, if the new transition is out of phase with the previous one, a few microns of erase band with no definite magnetization are created at the juncture of the old and new tracks. The write field in the erase band is above the coercivity of the media and would change the magnetization in these areas, but its magnitude is not high enough to create new well-defined transitions. One experiment involved writing a fixed pattern of all 1's with a bit interval of 2.5 µm, moving the write head off-track by approximately half a track width, and then writing the pattern again with a frequency slightly higher than that of the previously recorded track for a bit interval of 2.45 µm to create all possible phase differences between the transitions in the old and new tracks. Using a 4.2 µm wide head produced an erase band of approximately 1 µm in width when the old and new tracks were 180° out of phase, dropping to almost nothing when the two tracks were in-phase. Writing data at a higher frequency with the original tracks bit interval at 0.5 µm and the new tracks bit interval at 0.49 µm allows a single MFM image to contain all possible phase differences, showing a dramatic increase in the width of the erase band as the two tracks move from in-phase to 180° out of phase [10].

In addition, the new track width can exhibit modulation which depends on the phase relationship between the old and new patterns, allowing the previous data to be recovered even if the old data patterns themselves are no longer distinct. The overwrite performance also depends on the position of the write head relative to the originally written track. If the head is directly aligned with the track, overwrite performance is relatively good; as the head moves offtrack, the performance drops markedly as the remnant components of the original data are read back along with the newly-written signal. This effect is less noticeable as the write frequency increases due to the greater attenuation of the field with distance [11].

When all the above factors are combined it turns out that each track contains an image of everything ever written to it, but that the contribution from each "layer" gets progressively smaller the further back it was made. Intelligence organisations have a lot of expertise in recovering these palimpsestuous images.

# Erasure of Data stored on Magnetic Media

The general concept behind an overwriting scheme is to flip each magnetic domain on the disk back and forth as much as possible (this is the basic idea behind degaussing) without writing the same pattern twice in a row. If the data was encoded directly, we could simply choose the desired overwrite pattern of ones and zeroes and write it repeatedly. However, disks generally use some form of run-length limited (RLL) encoding, so that the adjacent ones won't be written. This encoding is used to ensure that transitions aren't placed too closely together, or too far apart, which would mean the drive would lose track of where it was in the data.

To erase magnetic media, we need to overwrite it many times with alternating patterns in order to expose it to a magnetic field oscillating fast enough that it does the desired flipping of the magnetic domains in a reasonable amount of time. Unfortunately, there is a complication in that we need to saturate the disk surface to the greatest depth possible, and very high frequency signals only "scratch the surface" of the magnetic medium. Disk drive manufacturers, in trying to achieve ever-higher densities, use the highest possible frequencies, whereas we really require the lowest frequency a disk drive can produce. Even this is still rather high. The best we can do is to use the lowest frequency possible for overwrites, to penetrate as deeply as possible into the recording medium.

The write frequency also determines how effectively previous data can be overwritten due to the dependence of the field needed to cause magnetic switching on the length of time the field is applied. Tests on a number of typical disk drive heads have shown a difference of up to 20 dB in overwrite performance when data recorded at 40 kFCI (flux changes per inch), typical of recent disk drives, is overwritten with a signal varying from 0 to 100 kFCI. The best average performance for the various heads appears to be with an overwrite signal of around 10 kFCI, with the worst performance being at 100 kFCI [12]. The track write width is also affected by the write frequency - as the frequency increases, the write width decreases for both MR and TFI heads. In [13] there was a decrease in write width of around 20% as the write frequency was increased from 1 to 40 kFCI, with the decrease being most marked at the high end of the frequency range. However, the decrease in write width is balanced by a corresponding increase in the two side- erase bands so that the sum of the two remains nearly constant with frequency and equal to the DC erase width for the head. The media coercivity also affects the width of the write and erase bands, with their width dropping as the coercivity increases (this is one of the explanations for the ever-increasing coercivity of newer, higher-density drives).

To try to write the lowest possible frequency we must determine what decoded data to write to produce a low-frequency encoded signal.

In order to understand the theory behind the choice of data patterns to write, it is necessary to take a brief look at the recording methods used in disk drives. The main limit on recording density is that as the bit density is increased, the peaks in the analog signal recorded on the media are read at a rate which may cause them to appear to overlap, creating intersymbol interference which leads to data errors. Traditional peak detector read channels try to reduce the possibility of intersymbol interference by coding data in such a way that the analog signal peaks are separated as far as possible. The read circuitry can then accurately detect the peaks (actually the head itself only detects transitions in magnetisation, so the simplest recording code uses a transition to encode a 1 and the absence of a transition to encode a 0. The transition causes a positive/negative peak in the head output voltage (thus the name "peak detector read channel"). To recover the data, we differentiate the output and look for the zero crossings). Since a long string of 0's will make clocking difficult, we need to set a limit on the maximum consecutive number of 0's. The separation of peaks is implemented as some form of run-length-limited, or RLL, coding.

The RLL encoding used in most current drives is described by pairs of run-length limits (*d, k*), where *d* is the minimum number of 0 symbols which must occur between each 1 symbol in the encoded data, and *k* is the maximum. The parameters (*d, k*) are chosen to place adjacent 1's far enough apart to avoid problems with intersymbol interference, but not so far apart that we lose synchronisation.

The grandfather of all RLL codes was FM, which wrote one user data bit followed by one clock bit, so that a 1 bit was encoded as two transitions (1 wavelength) while a 0 bit was encoded as one transition (« wavelength). A different approach was taken in modified FM (MFM), which suppresses the clock bit except between adjacent 0's

(the ambiguity in the use of the term MFM is unfortunate. From here on it will be used to refer to modified FM rather than magnetic force microscopy). Taking three example sequences 0000, 1111, and 1010, these will be encoded as 0(1)0(1)0(1)0, 1(0)1(0)1(0)1, and 1(0)0(0)1(0)0 (where the ()s are the clock bits inserted by the encoding process). The maximum time between 1 bits is now three 0 bits (so that the peaks are no more than four encoded time periods apart), and there is always at least one 0 bit (so that the peaks in the analog signal are at least two encoded time periods apart), resulting in a (1,3) RLL code. (1,3) RLL/MFM is the oldest code still in general use today, but is only really used in floppy drives which need to remain backwards-compatible.

These constraints help avoid intersymbol interference, but the need to separate the peaks reduces the recording density and therefore the amount of data which can be stored on a disk. To increase the recording density, MFM was gradually replaced by (2,7) RLL (the original "RLL" format), and that in turn by (1,7) RLL, each of which placed less constraints on the recorded signal.

Using our knowledge of how the data is encoded, we can now choose which decoded data patterns to write in order to obtain the desired encoded signal. The three encoding methods described above cover the vast majority of magnetic disk drives. However, each of these has several possible variants. With MFM, only one is used with any frequency, but the newest (1,7) RLL code has at least half a dozen variants in use. For MFM with at most four bit times between transitions, the lowest write frequency possible is attained by writing the repeating decoded data patterns 1010 and 0101. These have a 1 bit every other "data" bit, and the intervening "clock" bits are all 0. We would also like patterns with every other clock bit set to 1 and all others set to 0, but these are not possible in the MFM encoding (such "violations" are used to generate special marks on the disk to identify sector boundaries). The best we can do here is three bit times between transitions, which is generated by repeating the decoded patterns 100100, 010010 and 001001. We should use several passes with these patterns, as MFM drives are the oldest, lowest-density drives around (this is especially true for the very-low-density floppy drives). As such, they are the easiest to recover data from with modern equipment and we need to take the most care with them.

From MFM we jump to the next simplest case, which is (1,7) RLL. Although there can be as many as 8 bit times between transitions, the lowest sustained frequency we can have in practice is 6 bit times between transitions. This is a desirable property from the point of view of the clock-recovery circuitry, and all (1,7) RLL codes seem to have this property. We now need to find a way to write the desired pattern without knowing the particular (1,7) RLL code used. We can do this by looking at the way the drives error-correction system works. The error- correction is applied to the decoded data, even though errors generally occur in the encoded data. In order to make this work well, the data encoding should have limited error amplification, so that an erroneous encoded bit should affect only a small, finite number of decoded bits.

Decoded bits therefore depend only on nearby encoded bits, so that a repeating pattern of encoded bits will correspond to a repeating pattern of decoded bits. The repeating pattern of encoded bits is 6 bits long. Since the rate of the code is 2/3, this corresponds to a repeating pattern of 4 decoded bits. There are only 16 possibilities for this pattern, making it feasible to write all of them during the erase process. So to achieve good overwriting of (1,7) RLL disks, we write the patterns 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, and 1111. These patterns also conveniently cover two of the ones needed for MFM overwrites, although we should add a few more iterations of the MFM-specific patterns for the reasons given above.

Finally, we have (2,7) RLL drives. These are similar to MFM in that an eight-bit-time signal can be written in some phases, but not all. A six-bit-time signal will fill in the remaining cracks. Using a « encoding rate, an eight-bit-time signal corresponds to a repeating pattern of 4 data bits. The most common (2,7) RLL code is shown below:

**The most common (2,7) RLL Code**

| Decoded Data | (2,7) RLL Encoded Data |
|---|---|
| 00 | 1000 |
| 01 | 0100 |
| 100 | 001000 |
| 101 | 100100 |
| 111 | 000100 |
| 1100 | 00001000 |
| 1101 | 00100100 |

The second most common (2,7) RLL code is the same but with the "decoded data" complemented, which doesn't alter these patterns. Writing the required encoded data can be achieved for every other phase using patterns of 0x33, 0x66, 0xCC and 0x99, which are already written for (1,7) RLL drives.

Six-bit-time patterns can be written using 3-bit repeating patterns. The all-zero and all-one patterns overlap with the (1,7) RLL patterns, leaving six others:

```
001001001001001001001001
   2   4   9   2   4   9
```

in binary or 0x24 0x92 0x49, 0x92 0x49 0x24 and 0x49 0x24 0x92 in hex, and

```
011011011011011011011011
   6   D   B   6   D   B
```

in binary or 0x6D 0xB6 0xDB, 0xB6 0xDB 0x6D and 0xDB 0x6D 0xB6 in hex. The first three are the same as the MFM patterns, so we need only three extra patterns to cover (2,7) RLL drives.

Although (1,7) is more popular in recent (post-1990) drives, some older hard drives do still use (2,7) RLL, and with the ever-increasing reliability of newer drives it is likely that they will remain in use for some time to come, often being passed down from one machine to another. The above three patterns also cover any problems with endianness issues, which weren't a concern in the previous two cases, but would be in this case (actually, thanks to the strong influence of IBM mainframe drives, everything seems to be uniformly big-endian within bytes, with the most significant bit being written to the disk first).

The latest high-density drives use methods like Partial-Response Maximum-Likelihood (PRML) encoding, which may be roughly equated to the trellis encoding done by V.32 modems in that it is effective but computationally expensive. PRML codes are still RLL codes, but with somewhat different constraints. A typical code might have (0,4,4) constraints in which the 0 means that 1's in a data stream can occur right next to 0's (so that peaks in the analog readback signal are not separated), the first 4 means that there can be no more than four 0's between 1's in a data stream, and the second 4 specifies the maximum number of 0's between 1's in certain symbol subsequences. PRML codes avoid intersymbol influence errors by using digital filtering techniques to shape the read signal to exhibit desired frequency and timing characteristics (this is the "partial response" part of PRML) followed by maximum- likelihood digital data detection to determine the most likely sequence of data bits that was written to the disk (this is the "maximum likelihood" part of PRML). PRML channels achieve the same low bit error rate as standard peak-detection methods, but with much higher recording densities, while using the same heads and media. Several manufacturers are currently engaged in moving their peak-detection-based product lines across to PRML, giving a 30-40% density increase over standard RLL channels [14].

Since PRML codes don't try to separate peaks in the same way that non-PRML RLL codes do, all we can do is to write a variety of random patterns because the processing inside the drive is too complex to second- guess. Fortunately, these drives push the limits of the magnetic media much more than older drives ever did by encoding data with much smaller magnetic domains, closer to the physical capacity of the magnetic media (the current state of the art in PRML drives has a track density of around 6700 TPI (tracks per inch) and a data recording density of 170 kFCI, nearly double that of the nearest (1,7) RLL equivalent. A convenient side-effect of these very high recording densities is that a written transition may experience the write field cycles for successive transitions, especially at the track edges where the field distribution is much broader [15]. Since this is also where remnant data is most likely to be found, this can only help in reducing the recoverability of the data). If these drives require sophisticated signal processing just to read the most recently written data, reading overwritten layers is also correspondingly more difficult. A good scrubbing with random data will do about as well as can be expected.

We now have a set of 22 overwrite patterns which should erase everything, regardless of the raw encoding. The basic disk eraser can be improved slightly by adding random passes before and after the erase process, and by performing the deterministic passes in random order to make it more difficult to guess which of the known data passes were made at which point. To deal with all this in the overwrite process, we use the sequence of 35 consecutive writes shown below:

**Overwrite Data**

| Pass No. | Data Written | Encoding Scheme Targeted | | |
|---|---|---|---|---|
| 1 | Random | | | |
| 2 | Random | | | |
| 3 | Random | | | |
| 4 | Random | | | |
| 5 | 01010101 01010101 01010101 0x55 | (1,7) RLL | | MFM |
| 6 | 10101010 10101010 10101010 0xAA | (1,7) RLL | | MFM |
| 7 | 10010010 01001001 00100100 0x92 0x49 0x24 | | (2,7) RLL | MFM |
| 8 | 01001001 00100100 10010010 0x49 0x24 0x92 | | (2,7) RLL | MFM |
| 9 | 00100100 10010010 01001001 0x24 0x92 0x49 | | (2,7) RLL | MFM |
| 10 | 00000000 00000000 00000000 0x00 | (1,7) RLL | (2,7) RLL | |
| 11 | 00010001 00010001 00010001 0x11 | (1,7) RLL | | |
| 12 | 00100010 00100010 00100010 0x22 | (1,7) RLL | | |
| 13 | 00110011 00110011 00110011 0x33 | (1,7) RLL | (2,7) RLL | |
| 14 | 01000100 01000100 01000100 0x44 | (1,7) RLL | | |
| 15 | 01010101 01010101 01010101 0x55 | (1,7) RLL | | MFM |
| 16 | 01100110 01100110 01100110 0x66 | (1,7) RLL | (2,7) RLL | |
| 17 | 01110111 01110111 01110111 0x77 | (1,7) RLL | | |
| 18 | 10001000 10001000 10001000 0x88 | (1,7) RLL | | |
| 19 | 10011001 10011001 10011001 0x99 | (1,7) RLL | (2,7) RLL | |
| 20 | 10101010 10101010 10101010 0xAA | (1,7) RLL | | MFM |
| 21 | 10111011 10111011 10111011 0xBB | (1,7) RLL | | |
| 22 | 11001100 11001100 11001100 0xCC | (1,7) RLL | (2,7) RLL | |
| 23 | 11011101 11011101 11011101 0xDD | (1,7) RLL | | |
| 24 | 11101110 11101110 11101110 0xEE | (1,7) RLL | | |
| 25 | 11111111 11111111 11111111 0xFF | (1,7) RLL | (2,7) RLL | |
| 26 | 10010010 01001001 00100100 0x92 0x49 0x24 | | (2,7) RLL | MFM |
| 27 | 01001001 00100100 10010010 0x49 0x24 0x92 | | (2,7) RLL | MFM |
| 28 | 00100100 10010010 01001001 0x24 0x92 0x49 | | (2,7) RLL | MFM |
| 29 | 01101101 10110110 11011011 0x6D 0xB6 0xDB | | (2,7) RLL | |
| 30 | 10110110 11011011 01101101 0xB6 0xDB 0x6D | | (2,7) RLL | |
| 31 | 11011011 01101101 10110110 0xDB 0x6D 0xB6 | | (2,7) RLL | |
| 32 | Random | | | |
| 33 | Random | | | |
| 34 | Random | | | |
| 35 | Random | | | |

The MFM-specific patterns are repeated twice because MFM drives have the lowest density and are thus particularly easy to examine. The deterministic patterns between the random writes are permuted before the write is performed, to make it more difficult for an opponent to use knowledge of the erasure data written to attempt to recover overwritten data (in fact we need to use a cryptographically strong random number generator to perform the permutations to avoid the problem of an opponent who can read the last overwrite pass being able to predict the previous passes and "echo cancel" passes by subtracting the known overwrite data).

If the device being written to supports caching or buffering of data, this should be disabled to ensure that physical disk writes are performed for each pass instead of everything but the last pass being lost in the buffering. For example physical disk access can be forced during SCSI-2 Group 1 write commands by setting the Force Unit Access bit in the SCSI command block (although at least one popular drive has a bug which causes all writes to be ignored when this bit is set - remember to test your overwrite scheme before you deploy it). Another consideration which needs to be taken into account when trying to erase data through software is that drives conforming to some of the higher-level protocols such as the various SCSI standards are relatively free to interpret commands sent to them in whichever way they choose (as long as they still conform to the SCSI specification). Thus some drives, if sent a FORMAT UNIT command may return immediately without performing any action, may simply perform a read test on the entire disk (the most common option), or may actually write data to the disk (the SCSI- 2 standard includes an initialization pattern (IP) option for the FORMAT UNIT command, however this is not necessarily supported by existing drives).

If the data is very sensitive and is stored on floppy disk, it can best be destroyed by removing the media from the disk liner and burning it, or by burning the entire disk, liner and all (most floppy disks burn remarkably well - albeit with quantities of oily smoke - and leave very little residue).

# Other Methods of Erasing Magnetic Media

The previous section has concentrated on erasure methods which require no specialised equipment to perform the erasure. Alternative means of erasing media which do require specialised equipment are degaussing (a process in which the recording media is returned to its initial state) and physical destruction. Degaussing is a reasonably effective means of purging data from magnetic disk media, and will even work through most drive cases (research has shown that the aluminium housings of most disk drives attenuate the degaussing field by only about 2 dB [16]).

The switching of a single-domain magnetic particle from one magnetization direction to another requires the overcoming of an energy barrier, with an external magnetic field helping to lower this barrier. The switching depends not only on the magnitude of the external field, but also on the length of time for which it is applied. For typical disk drive media, the short-term field needed to flip enough of the magnetic domains to be useful in recording a signal is about 1/3 higher than the coercivity of the media (the exact figure varies with different media types) [17].

However, to effectively erase a medium to the extent that recovery of data from it becomes uneconomical requires a magnetic force of about five times the coercivity of the medium [18], although even small external magnetic fields are sufficient to upset the normal operation of a hard disk (typically a few gauss at DC, dropping to a few milligauss at 1 MHz). Coercivity (measured in Oersteds, Oe) is a property of magnetic material and is defined as the amount of magnetic field necessary to reduce the magnetic induction in the material to zero - the higher the coercivity, the harder it is to erase data from a medium. Typical figures for various types of magnetic media are given below:

**Typical Media Coercivity Figures**

| **Medium** | **Coercivity** |
|---|---|
| 5.25" 360K floppy disk | 300 Oe |
| 5.25" 1.2M floppy disk | 675 Oe |
| 3.5" 720K floppy disk | 300 Oe |
| 3.5" 1.44M floppy disk | 700 Oe |
| 3.5" 2.88M floppy disk | 750 Oe |
| 3.5" 21M floptical disk | 750 Oe |
| Older (1980's) hard disks | 900-1400 Oe |
| Newer (1990's) hard disks | 1400-2200 Oe |
| 1/2" magnetic tape | 300 Oe |
| 1/4" QIC tape | 550 Oe |
| 8 mm metallic particle tape | 1500 Oe |
| DAT metallic particle tape | 1500 Oe |

US Government guidelines class tapes of 350 Oe coercivity or less as low-energy or Class I tapes and tapes of 350-750 Oe coercivity as high-energy or Class II tapes. Degaussers are available for both types of tapes. Tapes of over 750 Oe coercivity are referred to as Class III, with no known degaussers capable of fully erasing them being known [19], since even the most powerful commercial AC degausser cannot generate the recommended 7,500 Oe needed for full erasure of a typical DAT tape currently used for data backups.

Degaussing of disk media is somewhat more difficult - even older hard disks generally have a coercivity equivalent to Class III tapes, making them fairly difficult to erase at the outset. Since manufacturers rate their degaussers in peak gauss and measure the field at a certain orientation which may not be correct for the type of medium being erased, and since degaussers tend to be rated by whether they erase sufficiently for clean rerecording rather than whether they make the information impossible to recover, it may be necessary to resort to physical destruction of the media to completely sanitise it (in fact since degaussing destroys the sync bytes, ID fields, error correction information, and other paraphernalia needed to identify sectors on the media, thus rendering the drive unusable, it makes the degaussing process mostly equivalent to physical destruction). In addition, like physical destruction, it requires highly specialised equipment which is expensive and difficult to obtain (one example of an adequate degausser was the 2.5 MW Navy research magnet used by a former Pentagon site manager to degauss a 14" hard drive for 1« minutes. It bent the platters on the drive and probably succeeded in erasing it beyond the capabilities of

any data recovery attempts [20]).

# Further Problems with Magnetic Media

A major issue which cannot be easily addressed using any standard software-based overwrite technique is the problem of defective sector handling. When the drive is manufactured, the surface is scanned for defects which are added to a defect list or flaw map. If further defects, called grown defects, occur during the life of the drive, they are added to the defect list by the drive or by drive management software. There are several techniques which are used to mask the defects in the defect list. The first, alternate tracks, moves data from tracks with defects to known good tracks. This scheme is the simplest, but carries a high access cost, as each read from a track with defects requires seeking to the alternate track and a rotational latency delay while waiting for the data location to appear under the head, performing the read or write, and, if the transfer is to continue onto a neighbouring track, seeking back to the original position. Alternate tracks may be interspersed among data tracks to minimise the seek time to access them.

A second technique, alternate sectors, allocates alternate sectors at the end of the track to minimise seeks caused by defective sectors. This eliminates the seek delay, but still carries some overhead due to rotational latency. In addition it reduces the usable storage capacity by 1-3%.

A third technique, inline sector sparing, again allocates a spare sector at the end of each track, but resequences the sector ID's to skip the defective sector and include the spare sector at the end of the track, in effect pushing the sectors past the defective one towards the end of the track. The associated cost is the lowest of the three, being one sector time to skip the defective sector [21].

The handling of mapped-out sectors and tracks is an issue which can't be easily resolved without the cooperation of hard drive manufacturers. Although some SCSI and IDE hard drives may allow access to defect lists and even to mapped-out areas, this must be done in a highly manufacturer- and drive-specific manner. For example the SCSI-2 READ DEFECT DATA command can be used to obtain a list of all defective areas on the drive. Since SCSI logical block numbers may be mapped to arbitrary locations on the disk, the defect list is recorded in terms of heads, tracks, and sectors. As all SCSI device addressing is performed in terms of logical block numbers, mapped-out sectors or tracks cannot be addressed. The only reasonably portable possibility is to clear various automatic correction flags in the read-write error recovery mode page to force the SCSI device to report read/write errors to the user instead of transparently remapping the defective areas. The user can then use the READ LONG and WRITE LONG commands (which allow access to sectors and extra data even in the presence of read/write errors), to perform any necessary operations on the defective areas, and then use the REASSIGN BLOCKS command to reassign the defective sections. However this operation requires an in-depth knowledge of the operation of the SCSI device and extensive changes to disk drivers, and more or less defeats the purpose of having an intelligent peripheral.

The ANSI X3T-10 and X3T-13 subcommittees are currently looking at creating new standards for a Universal Security Reformat command for IDE and SCSI hard disks which will address these issues. This will involve a multiple-pass overwrite process which covers mapped-out disk areas with deliberate off-track writing. Many drives available today can be modified for secure erasure through a firmware upgrade, and once the new firmware is in place the erase procedure is handled by the drive itself, making unnecessary any interaction with the host system beyond the sending of the command which begins the erase process.

Long-term ageing can also have a marked effect on the erasability of magnetic media. For example, some types of magnetic tape become increasingly difficult to erase after being stored at an elevated temperature or having contained the same magnetization pattern for a considerable period of time [22]. The same applies for magnetic disk media, with decreases in erasability of several dB being recorded [23]. The erasability of the data depends on the amount of time it has been stored on the media, not on the age of the media itself (so that, for example, a five-year-old freshly-written disk is no less erasable than a new freshly-written disk).

The dependence of media coercivity on temperature can affect overwrite capability if the data was initially recorded at a temperature where the coercivity was low (so that the recorded pattern penetrated deep into the media), but must be overwritten at a temperature where the coercivity is relatively high. This is important in hard disk drives, where the temperature varies depending on how long the unit has been used and, in the case of drives with power-saving features enabled, how recently and frequently it has been used. However the overwrite performance depends not

only on temperature-dependent changes in the media, but also on temperature-dependent changes in the read/write head. Thankfully the combination of the most common media used in current drives with various common types of read/write heads produce a change in overwrite performance of only a few hundredths of a decibel per degree over the temperature range -40°C to + 40°C, as changes in the head compensate for changes in the media [24].

Another issue which needs to be taken into account is the ability of most newer storage devices to recover from having a remarkable amount of damage inflicted on them through the use of various error-correction schemes. As increasing storage densities began to lead to multiple-bit errors, manufacturers started using sophisticated error-correction codes (ECC's) capable of correcting multiple error bursts. A typical drive might have 512 bytes of data, 4 bytes of CRC, and 11 bytes of ECC per sector. This ECC would be capable of correcting single burst errors of up to 22 bits or double burst errors of up to 11 bits, and can detect a single burst error of up to 51 bits or three burst errors of up to 11 bits in length [25]. Another drive manufacturer quotes the ability to correct up to 120 bits, or up to 32 bits on the fly, using 198-bit Reed-Solomon ECC [26]. Therefore even if some data is reliably erased, it may be possible to recover it using the built-in error-correction capabilities of the drive. Conversely, any erasure scheme which manages to destroy the ECC information (for example through the use of the SCSI-2 WRITE LONG command which can be used to write to areas of a disk sector outside the normal data areas) stands a greater chance of making the data unrecoverable.

# Sidestepping the Problem

The easiest way to solve the problem of erasing sensitive information from magnetic media is to ensure that it never gets to the media in the first place. Although not practical for general data, it is often worthwhile to take steps to keep particularly important information such as encryption keys from ever being written to disk. This would typically happen when the memory containing the keys is paged out to disk by the operating system, where they can then be recovered at a later date, either manually or using software which is aware of the in-memory data format and can locate it automatically in the swap file (for example there exists software which will search the Windows swap file for keys from certain DOS encryption programs). An even worse situation occurs when the data is paged over a network, allowing anyone with a packet sniffer or similar tool on the same subnet to observe the information (for example there exists software which will monitor and even alter NFS traffic on the fly which could be modified to look for known in-memory data patterns moving to and from a networked swap disk [27]).

To solve these problems the memory pages containing the information can be locked to prevent them from being paged to disk or transmitted over a network. This approach is taken by at least one encryption library, which allocates all keying information inside protected memory blocks visible to the user only as opaque handles, and then optionally locks the memory (provided the underlying OS allows it) to prevent it from being paged [28]. The exact details of locking pages in memory depend on the operating system being used. Many Unix systems now support the `mlock()`/`munlock()` calls or have some alternative mechanism hidden among the `mmap()`-related functions which can be used to lock pages in memory. Unfortunately these operations require superuser privileges because of their potential impact on system performance if large ranges of memory are locked. Other systems such as Microsoft Windows NT allow user processes to lock memory with the `VirtualLock()`/`VirtualUnlock()` calls, but limit the total number of regions which can be locked.

Most paging algorithms are relatively insensitive to having sections of memory locked, and can even relocate the locked pages (since the logical to physical mapping is invisible to the user), or can move the pages to a "safe" location when the memory is first locked. The main effect of locking pages in memory is to increase the minimum working set size which, taken in moderation, has little noticeable effect on performance. The overall effects depend on the operating system and/or hardware implementations of virtual memory. Most Unix systems have a global page replacement policy in which a page fault may be satisfied by any page frame. A smaller number of operating systems use a local page replacement policy in which pages are allocated from a fixed (or occasionally dynamically variable) number of page frames allocated on a per- process basis. This makes them much more sensitive to the effects of locking pages, since every locked page decreases the (finite) number of pages available to the process. On the other hand it makes the system as a whole less sensitive to the effects of one process locking a large number of pages. The main effective difference between the two is that under a local replacement policy a process can only lock a small fixed number of pages without affecting other processes, whereas under a global replacement policy the number of pages a process can lock is determined on a system-wide basis and may be affected by other processes.

In practice neither of these allocation strategies seem to cause any real problems. Although any practical measurements are very difficult to perform since they vary wildly depending on the amount of physical memory present, paging strategy, operating system, and system load, in practice locking a dozen 1K regions of memory (which might be typical of a system on which a number of users are running programs such as mail encryption software) produced no noticeable performance degradation observable by system- monitoring tools. On machines such as network servers handling large numbers of secure connections (for example an HTTP server using SSL), the effects of locking large numbers of pages may be more noticeable.

# Methods of Recovery for Data stored in Random-Access Memory

Contrary to conventional wisdom, "volatile" semiconductor memory does not entirely lose its contents when power is removed. Both static (SRAM) and dynamic (DRAM) memory retains some information on the data stored in it while power was still applied. SRAM is particularly susceptible to this problem, as storing the same data in it over a long period of time has the effect of altering the preferred power-up state to the state which was stored when power was removed. Older SRAM chips could often "remember" the previously held state for several days. In fact, it is possible to manufacture SRAM's which always have a certain state on power-up, but which can be overwritten later on - a kind of "writeable ROM".

DRAM can also "remember" the last stored state, but in a slightly different way. It isn't so much that the charge (in the sense of a voltage appearing across a capacitance) is retained by the RAM cells, but that the thin oxide which forms the storage capacitor dielectric is highly stressed by the applied field, or is not stressed by the field, so that the properties of the oxide change slightly depending on the state of the data. One thing that can cause a threshold shift in the RAM cells is ionic contamination of the cell(s) of interest, although such contamination is rarer now than it used to be because of robotic handling of the materials and because the purity of the chemicals used is greatly improved. However, even a perfect oxide is subject to having its properties changed by an applied field. When it comes to contaminants, sodium is the most common offender - it is found virtually everywhere, and is a fairly small (and therefore mobile) atom with a positive charge. In the presence of an electric field, it migrates towards the negative pole with a velocity which depends on temperature, the concentration of the sodium, the oxide quality, and the other impurities in the oxide such as dopants from the processing. If the electric field is zero and given enough time, this stress tends to dissipate eventually.

The stress on the cell is a cumulative effect, much like charging an RC circuit. If the data is applied for only a few milliseconds then there is very little "learning" of the cell, but if it is applied for hours then the cell will acquire a strong (relatively speaking) change in its threshold. The effects of the stress on the RAM cells can be measured using the built-in self test capabilities of the cells, which provide the ability to impress a weak voltage on a storage cell in order to measure its margin. Cells will show different margins depending on how much oxide stress has been present. Many DRAM's have undocumented test modes which allow some normal I/O pin to become the power supply for the RAM core when the special mode is active. These test modes are typically activated by running the RAM in a nonstandard configuration, so that a certain set of states which would not occur in a normally-functioning system has to be traversed to activate the mode. Manufacturers won't admit to such capabilities in their products because they don't want their customers using them and potentially rejecting devices which comply with their spec sheets, but have little margin beyond that.

A simple but somewhat destructive method to speed up the annihilation of stored bits in semiconductor memory is to heat it. Both DRAM's and SRAM's will lose their contents a lot more quickly at Tjunction = 140°C than they will at room temperature. Several hours at this temperature with no power applied will clear their contents sufficiently to make recovery difficult. Conversely, to extend the life of stored bits with the power removed, the temperature should be dropped below -60°C. Such cooling should lead to weeks, instead of hours or days, of data retention.

# Erasure of Data stored in Random-Access Memory

Simply repeatedly overwriting the data held in DRAM with new data isn't nearly as effective as it is for magnetic media. The new data will begin stressing or relaxing the oxide as soon as it is written, and the oxide will immediately begin to take a "set" which will either reinforce the previous "set" or will weaken it. The greater the amount of time that new data has existed in the cell, the more the old stress is "diluted", and the less reliable the information extraction will be. Generally, the rates of change due to stress and relaxation are in the same order of magnitude. Thus, a few microseconds of storing the opposite data to the currently stored value will have little effect on the oxide. Ideally, the oxide should be exposed to as much stress at the highest feasible temperature and for as long as possible to get the greatest "erasure" of the data. Unfortunately if carried too far this has a rather detrimental effect on the life expectancy of the RAM.

Therefore the goal to aim for when sanitising memory is to store the data for as long as possible rather than trying to change it as often as possible. Conversely, storing the data for as short a time as possible will reduce the chances of it being "remembered" by the cell. Based on tests on DRAM cells, a storage time of one second causes such a small change in threshold that it probably isn't detectable. On the other hand, one minute is probably detectable, and 10 minutes is certainly detectable.

The most practical solution to the problem of DRAM data retention is therefore to constantly flip the bits in memory to ensure that a memory cell never holds a charge long enough for it to be "remembered". While not practical for general use, it is possible to do this for small amounts of very sensitive data such as encryption keys. This is particularly advisable where keys are stored in the same memory location for long periods of time and control access to large amounts of information, such as keys used for transparent encryption of files on disk drives. The bit-flipping also has the convenient side-effect of keeping the page containing the encryption keys at the top of the queue maintained by the system's paging mechanism, greatly reducing the chances of it being paged to disk at some point.

# Conclusion

Data overwritten once or twice may be recovered by subtracting what is expected to be read from a storage location from what is actually read. Data which is overwritten an arbitrarily large number of times can still be recovered provided that the new data isn't written to the same location as the original data (for magnetic media), or that the recovery attempt is carried out fairly soon after the new data was written (for RAM). For this reason it is effectively impossible to sanitise storage locations by simple overwriting them, no matter how many overwrite passes are made or what data patterns are written. However by using the relatively simple methods presented in this paper the task of an attacker can be made significantly more difficult, if not prohibitively expensive.

# Acknowledgments

The author would like to thank Nigel Bree, Peter Fenwick, Andy Hospodor, Kevin Martinez, Colin Plumb, and Charles Preston for their advice and input during the preparation of this paper.

# References

[1] "Emergency Destruction of Information Storing Media", M.Slusarczuk et al, Institute for Defense Analyses, December 1987.
[2] "A Guide to Understanding Data Remanence in Automated Information Systems", National Computer Security Centre, September 1991.
[3] "Detection of Digital Information from Erased Magnetic Disks", Venugopal Veeravalli, Masters thesis, Carnegie-Mellon University, 1987.
[4] "Magnetic force microscopy: General principles and application to longitudinal recording media", D.Rugar, H.Mamin, P.Guenther, S.Lambert, J.Stern, I.McFadyen, and T.Yogi, *Journal of Applied Physics*, **Vol.68, No.3** (August 1990), p.1169.
[5] "Tunneling-stabilized Magnetic Force Microscopy of Bit Tracks on a Hard Disk", Paul Rice and John Moreland, *IEEE Trans.on Magnetics*, **Vol.27, No.3** (May 1991), p.3452.
[6] "NanoTools: The Homebrew STM Page", Jim Rice, NanoTools: The Homebrew STM Page, http://www.skypoint.com/members/jrice/STMWebPage.html.
[7] "Magnetic Force Scanning Tunnelling Microscope Imaging of Overwritten Data", Romel Gomez, Amr Adly, Isaak Mayergoyz, Edward Burke, *IEEE Trans.on Magnetics*, **Vol.28, No.5** (September 1992), p.3141.
[8] "Comparison of Magnetic Fields of Thin-Film Heads and Their Corresponding Patterns Using Magnetic Force Microscopy", Paul Rice, Bill Hallett, and John Moreland, *IEEE Trans.on Magnetics*, **Vol.30, No.6** (November 1994), p.4248.
[9] "Computation of Magnetic Fields in Hysteretic Media", Amr Adly, Isaak Mayergoyz, Edward Burke, *IEEE Trans.on Magnetics*, **Vol.29, No.6** (November 1993), p.2380.
[10] "Magnetic Force Microscopy Study of Edge Overwrite Characteristics in Thin Film Media", Jian- Gang Zhu, Yansheng Luo, and Juren Ding, *IEEE Trans.on Magnetics*, **Vol.30, No.6** (November 1994), p.4242.
[11] "Microscopic Investigations of Overwritten Data", Romel Gomez, Edward Burke, Amr Adly, Isaak Mayergoyz, J.Gorczyca, *Journal of Applied Physics*, **Vol.73, No.10** (May 1993), p.6001.
[12] "Relationship between Overwrite and Transition Shift in Perpendicular Magnetic Recording", Hiroaki Muraoka, Satoshi Ohki, and Yoshihisa Nakamura, *IEEE Trans.on Magnetics*, **Vol.30, No.6** (November 1994), p.4272.
[13] "Effects of Current and Frequency on Write, Read, and Erase Widths for Thin-Film Inductive and Magnetoresistive Heads", Tsann Lin, Jodie Christner, Terry Mitchell, Jing-Sheng Gau, and Peter George, *IEEE Trans.on Magnetics*, **Vol.25, No.1** (January 1989), p.710.
[14] "PRML Read Channels: Bringing Higher Densities and Performance to New-Generation Hard Drives", Quantum Corporation, 1995.
[15] "Density and Phase Dependence of Edge Erase Band in MR/Thin Film Head Recording", Yansheng Luo, Terence Lam, Jian-Gang Zhu, *IEEE Trans.on Magnetics*, **Vol.31, No.6** (November 1995), p.3105.
[16] "A Guide to Understanding Data Remanence in Automated Information Systems", National Computer Security Centre, September 1991.
[17] "Time-dependant Magnetic Phenomena and Particle-size Effects in Recording Media", *IEEE Trans.on Magnetics*, **Vol.26, No.1** (January 1990), p.193.
[18] "The Data Dilemna", Charles Preston, Security Management Journal, February 1995.
[19] "Magnetic Tape Degausser", NSA/CSS Specification L14-4-A, 31 October 1985.
[20] "How many times erased does DoD want?", David Hayes, posting to comp.periphs.scsi newsgroup, 24 July 1991, message-ID 1991Jul24.050701.16005@sulaco.lone star.org.
[21] "The Changing Nature of Disk Controllers", Andrew Hospodor and Albert Hoagland, *Proceedings of the IEEE*, **Vol.81, No.4** (April 1993), p.586.
[22] "Annealing Study of the Erasability of High Energy Tapes", L.Lekawat, G.Spratt, and M.Kryder, *IEEE Trans.on Magnetics*, **Vol.29, No.6** (November 1993), p.3628.
[23] "The Effect of Aging on Erasure in Particulate Disk Media", K.Mountfield and M.Kryder, *IEEE Trans.on Magnetics*, **Vol.25, No 5** (September 1989), p.3638.
[24] "Overwrite Temperature Dependence for Magnetic Recording", Takayuki Takeda, Katsumichi Tagami, and Takaaki Watanabe, *Journal of Applied Physics*, **Vol.63, No.8** (April 1988), p.3438.
[25] Conner 3.5" hard drive data sheets, 1994, 1995.
[26] "Technology and Time-to-Market: The Two Go Hand-in-Hand", Quantum Corporation, 1995.
[27] "Basic Flaws in Internet Security and Commerce", Paul Gauthier, posting to comp.security.unix newsgroup, 9 October 1995, message-ID gauthier.813274073@espresso.cs.ber keley.edu.
[28] "cryptlib Free Encryption Library", Peter Gutmann, cryptlib, http://www.cs.auckland.ac.nz/~pgut001/cryptlib.html.

## Author

You can contact the author via Internet e-mail,
sami.tolvanen@iki.fi.

If you are having problems with this software or have a suggestion which you think would improve this product, do not hesitate to e-mail me.

---

**See also:**
**Obtaining the latest version**

## Obtaining the Latest Version

The latest version of Eraser can be found at World Wide Web home page,
http://www.iki.fi/st/eraser/.

**Note!** The page is not physically located at domain iki.fi, but you will be redirected to the right location. The physical location (server) of this page is subject to change, so if you want to link to Eraser home page, please use the URL above. Thank you.

**Eraser Home Page**
http://www.iki.fi/st/eraser/