

# md5deep and hashdeep - Latest version 4.0.1

---

## Quick Links

- [Download md5deep and hashdeep](#)
- Manual pages for [md5deep](#), [hashdeep](#)
- Getting started guides for [md5deep](#), [hashdeep](#)
- [Donate to the project](#)
- [Sourceforge project page](#) - Home to development and feature requests

## Introduction

**md5deep** is a set of programs to compute [MD5](#), [SHA-1](#), [SHA-256](#), [Tiger](#), or [Whirlpool](#) message digests on an arbitrary number of files. md5deep is similar to the md5sum program found in the [GNU Coreutils](#) package, but has the following additional features:

- Recursive operation - md5deep is able to recursively examine an entire directory tree. That is, compute the MD5 for every file in a directory and for every file in every subdirectory.
- Comparison mode - md5deep can accept a list of known hashes and compare them to a set of input files. The program can display either those input files that match the list of known hashes or those that do not match. Hashes sets can be drawn from [Encase](#), the [National Software Reference Library](#), [iLook Investigator](#), [Hashkeeper](#), [md5sum](#), BSD md5, and other generic hash generating programs. Users are welcome to add functionality to read other formats too!
- Time estimation - md5deep can produce a time estimate when it's processing very large files.
- Piecewise hashing - Hash input files in arbitrary sized blocks
- File type mode - md5deep can process only files of a certain type, such as regular files, block devices, etc.

**hashdeep** is a program to compute, match, and *audit* hashsets. With traditional matching, programs report if an input file matched one in a set of knowns or if the input file did not match. It's hard to get a complete sense of the state of the input files compared to the set of knowns. It's possible to have matched files, missing files, files that have moved in the set, and to find new files not in the set. Hashdeep can report all of these conditions. It can even spot hash collisions, when an input file matches a known file in one hash algorithm but not in others. The results are displayed in an audit report.

The hash set audit process was described in the paper [Auditing Hash Sets: Lessons Learned from Jurassic Park](#) published in the journal [Digital Investigation](#).

For more details and some examples of using hashdeep, check out the [hashdeep getting started guide](#).

## Supported Platforms

The programs are distributed as binaries for Microsoft Windows (7, Vista, XP, 2003, and 2000 are supported) and as source code. The source code should compile nicely on just about any platform, including Cygwin, Linux, FreeBSD, OpenBSD, Mac OS X, OpenSolaris, HP/UX, etc.

Note that several operating systems support an automatic installation of md5deep through packaging or ports. See the [Automatic Installation](#) section of the getting started guide for details.

Microsoft Windows NT and Windows 9x are no longer supported. The last version that works on Windows 9x was version 1.13. Windows 3.x was never supported.

## Download md5deep and hashdeep

### Stable Version

The latest stable versions of md5deep and hashdeep are version 4.0.1 and was released on 22 Jan 2012. You can take a look at the [complete changelog](#), but here are the changes in the latest version:

### Bug Fixes

- Fixed hang on DFXML generation on Win32
- Fixed incorrect hashes via stdin on Win32
- Fixed "Too many open files" error on OS X
- Doc files in Win32 have been corrected.

Version 4.0.1	22 Jan 2012	<a href="#">Windows binary</a>	SHA256 b6deb5c038b85513f6d9a0ed64fc4d58af0c1fe3de34eb040aa0cd8434f9e475
		<a href="#">source code</a>	SHA256 79017543fa88c61700874cc7ed3c621069d2ac499d8d1aa145a2be9558e707de

### Beta Version

There is no beta version of md5deep right now. If you have any problems or would like to see something added to md5deep, please send mail to the developer at [research@jessekornblum.com](mailto:research@jessekornblum.com) or visit the [Sourceforge project page](#).

### Older Versions

Although older versions of md5deep and hashdeep are available for historical purposes, you shouldn't use these unless you have a truly compelling reason.

[Show older versions](#)

## The Hashing Algorithms

### MD5

To quote the executive summary of [RFC 1321, the official MD5 specification](#):

[MD5] takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest.

Some weaknesses have been found in MD5 and the above statement may no longer be accurate. Please check [the current status of MD5](#) for more information.

The algorithm was originally developed by Professor [Ron Rivest](#) of the Massachusetts Institute of Technology. More information on the algorithm can be found in [RFC 1321](#) and in the [RSA Cryptography FAQ](#).

### SHA-1

To quote the start of [RFC 3174](#):

[SHA-1 is used] for computing a condensed representation of a message or a data file. When a message of any length  $< 2^{64}$  bits is input, the SHA-1 produces a 160-bit output called a message digest.

The algorithm was originally developed by the [National Institute of Standards and Technology](#) and is officially defined in [FIPS 180-1](#), published in 1995. (The SHA-1 algorithm also appears in [FIPS 180-2](#), published in 2002. Although the algorithm was not changed, some of the notation was changed to make it more consistent with other algorithms.)

Some weaknesses have been found in SHA-1 and the above statements may no longer be accurate. Please check [the current status of SHA-1](#) for more information.

### SHA-256

To quote the start of [FIPS 180-2](#):

This Standard specifies ... secure hash algorithms ... for computing a condensed representation of electronic data (message). When a message of any length  $< 2^{64}$  bits is input to [SHA-256], the result is an output called a message digest.

The algorithm was originally developed by the [National Institute of Standards and Technology](#) and is officially defined in [FIPS 180-2](#), published in 2002.

## Tiger

Tiger is a hashing function developed in 1995 by Ross Anderson and Eli Biham. The algorithm produces a 192 bit output, but truncated versions of 160 and 128 bits, called Tiger/160 and Tiger/128 also exist. A second version of the algorithm, Tiger2, is identical to the original except for the padding method. The padding in Tiger2 is the same as MD5 and SHA-1. You can read more about Tiger at [the official Tiger website](#).

Tigerdeep, the program included in this package, produces Tiger/192 hashes in the same format as the NESSIE standard. Use the [NESSIE formatted test vectors](#) for Tiger to check your version of tigerdeep.

## Whirlpool

According to the [official Whirlpool homepage](#),

Whirlpool is a hash function designed by Vincent Rijmen and Paulo S. L. M. Barreto that operates on messages less than  $2^{256}$  bits in length, and produces a message digest of 512 bits. ... the final version (called simply WHIRLPOOL for short) was adopted by the International Organization for Standardization (ISO) in the [ISO/IEC 10118-3:2004](#) standard.

## About the developer

These programs were written and are now maintained by [Jesse Kornblum](#). At the time md5deep was originally written, Mr. Kornblum was a Special Agent with the [United States Air Force Office of Special Investigations](#) (AFOSI). As such, the program is considered to be a work of the United States Government. Under [17 USC 105](#), works of the United States Government are not eligible for copyright protection.

Please send all correspondence to [research@jessekornblum.com](mailto:research@jessekornblum.com).

## Acknowledgements

The developer wishes to thank Andreas Bussjaeger, Peter Chuenkov (aka ODB), Jeff Bryner, Matt Kucenski, Derek Jones, Daniel B. Sedory, Christopher T. Beers, Christophe Devine, Brian Carrier, Matt Harris, Derrick Donnelly, Matt Johnson, Dewayne Duff, Kris Kendall, Paul Alvarez, Ray Gagne, and Ed Kong. The testing of this program was made possible in part thanks to the generosity of the Computer Science Department at the University of Iowa.

This page was last updated on 22 Jan 2012.

---

