Checksum Programs Evaluation

Summary

Being able to verify that a file has not changed overtime at the bit level can be done by creating and monitoring the fixity of a file. Fixity can be generated on the server side, and there are also various types of programs available to assist with this task.

This paper provides basic information about checksums and describes the initial experience of five different easily accessible programs.

DISCLAIMER:
This is a topical overview and nowise intended to offer legal advice. Consult an attorney for assistance with specific concerns or for advice.

Any comments, corrections, or recommendations may be sent to the project team, care of:

Carol Kussmann
Digital Collections Assistant
Minnesota State Archives / Minnesota Historical Society
carol.kussmann@mnhs.org / 651.259.3262

Introduction to Checksums

When addressing digital preservation, one of the underlying concepts is that you must be able to authentication the content. You must be able to demonstrate that the content has not changed over time or during a transfer.

One of the simplest methods to do this is to use checksums. A checksum is an algorithm run on a document whose resulting number is referred to as the checksum value. The checksum value is a unique number assigned to a document based on the document's content at the bit level. After this checksum value is known, a checksum can be re-calculated on the document again at any time. Comparing checksum values, if the numbers are identical then the document has not changed; if the same number was not generated then the document has changed somehow. If differences are detected, this method does not tell you what has changed or when something was changed, just that the two documents are no longer identical at the bit level.

For example, if you need to send a document to someone else and want to ensure that the document they received is exactly the same as what you sent, you could apply a hash function (the mathematical algorithm) to the document, resulting in the checksum value. The person who received the document could then run a checksum to see if the checksum value on the document they received is the same. If the checksums do not match, the document is not the same as what was sent.

There are many different checksum algorithms, with the most common belonging to the MD5 and SHA families. Others include various algorithms belonging to the RIPEMD and TIGER families. Not all checksum methods are created equally; the easier to create, the easier it is to hack and use for possibly malicious reasons. Improvements and new methods are always being developed.

A known vulnerability of checksum algorithms is hash collisions. A hash collision is when two different bit streams (files) produce the same checksum value. Although very rare, both MD5 and SHA-1 have been found to do this by chance. Of more concern is when file checksums are manipulated and repeated on purpose for malicious reasons. Because of this, MD5 is specifically not recommended for use in United States government offices[1] and the National Institute of Standards and Technology (NIST) recommends not using SHA-1 for certain transactions[2]. MD5 and SHA-1 however may still be very appropriate for your internal use and are very common in many circumstances. Evaluate your use cases in order to pick the best checksum algorithm to use. As they are still widely in use, the programs below mainly focus on the MD5 and SHA-1 methods; a few include additional algorithms.

Program Evaluations

Goal: Evaluate programs that generate checksum values and a report in order to prepare content for transfer to an offsite repository and be able to verify that what the repository receives is what was sent.

Programs: The programs evaluated were found on the Web and included FastSum, ExactFile, HashMyFiles, the File Checksum Integrity Verifier Utility (for Windows), and Checksums for Windows. [Although there are other programs out there, this was a sampling of easily accessible tools that could be run from a desktop. There is no single place to go to find checksum programs.]

General Evaluation: After installing all of the programs on my computer I began to see how they fit my needs. I wanted to be able to create checksum values and save them in a report that could be sent with my files to the offsite repository. Some programs easily fit my needs, while others did not. Overall documentation on the programs varied.

FastSum

FastSum can both calculate and verify checksum values using a wizard that walks you through the process or by using the application itself. Checksums can be run on individual files, the contents of single folders, and the content of nested folders. Checksums use the MD5 algorithm only.

[1] United States Computer Emergency Readiness Team (US-CERT). *Vulnerability Note VU#836068.* January 21, 2009. http://www.kb.cert.org/vuls/id/836068

[2] National Institute of Standards and Technology (NIST). *NIST's Policy on Hash Functions.* March 15, 2006. http://csrc.nist.gov/groups/ST/hash/policy.html

The resulting checksum values can be saved in a Checksum List. The Checksum List includes at a minimum the checksum value and the file name; other properties can be added to the report by editing the settings. Checksum lists can be created and saved as one per file, one per folder, or one per root (top level) folder. This adds flexibility to the reuse of the checksum values.

ExactFile

ExactFile will let you calculate checksums on individual files as well as on a group of files. ExactFile can calculate checksums with many different algorithms including those in the MD, SHA, RIPEMD and TIGER families. Checksum values can be calculated with multiple methods at one time on single files, while one method must be selected when creating a digest (checksums on multiple files).

If you calculate checksums on a set of nested folders, the resulting checksums will be saved in a file in the top level folder, rather than with the individual files. This file is saved with a file extension for the method being used.

This program can also be used to verify checksums. Any file with a hash method extension (.md5, .sha1) can be opened with this program and can be used to verify the checksums. (This program opened the Checksum Lists created by FastSum and simply verified the checksums.) Simply open the file, point to the files, and click run. Reports will identify any problems.

Another feature of this program is that it allows you to create a TestFile Applet. This Applet was created for use when burning files to CD, but also works well on folders on servers. A digest is created that lists the checksums for multiple files, but it also creates at TestFile.exe file that can be run from the CD or folder in which it is in. This allows users without the ExactFile program to make sure the files have not changed.

There is also a command line tool available for this program.

HashMyFiles

HashMyFiles is a program that supports multiple checksum methods including MD5, CRC32, SHA-1, SHA-256, SHA-512, and SHA-384. Hash values are immediately calculated upon file selection.

Files on which checksums are to be run can be added in a variety of ways, however there is only one way to add content that includes nested folders.

Reports include the file name, hash values of the chosen methods, the full path of the file, modified time of the file, created time of the file, file size, file version, product version, identical, extension, and file attribute. Reports however are not automatically saved. To save the information, files must be selected. This allows the flexibility of saving all, none, or part of the information in the report.

Reports can be saved as a text file (.txt), a HTML file (.htm, .html), an XML file (.xml), or a comma delimited file (.csv). Unlike with some other programs that automatically save reports in the same location as the hashed files, you need to choose where to store the reports.

This program has additional features that color codes identical hash values in a report, allows you to customize the report columns, and displays individual file properties in a new window if desired.

This program can also be run from the command line on files and folder as well as from within a Windows Explorer menu on individual files.

File Checksum Integrity Verifier Utility
This is a command line utility that supports MD5 and SHA1 checksum algorithms. MD5 is the default method.

Using the command line in conjunction with a file structure that is part of a larger organization is very difficult. The command line does not allow spaces in file names. This is very difficult in a shared Windows environment where the folders are automatically created and named for you such as My Documents and My Pictures. To use this application, everything had to be moved to my local computer which was not very practical. Because of this, I did not explore this program any further; however, I was able to create MD5 hash values on files stored on my hard drive.

Checksums for Windows
If you are looking for a program to assist you with verifying checksum values on an individual file basis, CheckSums for Windows provides this simple service for MD5, SHA-1, SHA-256, SHA-384, and SHA-512. You can calculate and verify checksums for each of these algorithms one file at a time. This program does not do anything in batch format and does not save the values for you.

Summary

These five programs are just a sampling of easily accessible tools available for creating and verifying checksums. To find other available tools it is best to perform Internet searches to narrow down your results based on your criteria.

More details on the testing of all of these programs including screen shots can be found in the Minnesota State Archive's Center for Archival Resources On Legislatures[3].

[3] Minnesota State Archives. *Preservation Tools.* Center for Archival Resources On Legislatures (CAROL). May 2012. http://www.mnhs.org/preserve/records/legislativerecords/carol/preservation.htm#tools

Resources

General

Bailey, Jefferson. *File Fixity and Digital Preservation Storage: More Results from the NDSA Storage Survey*. The Signal, Digital Preservation Blog. March 6, 2012. http://blogs.loc.gov/digitalpreservation/2012/03/file-fixity-and-digital-preservation-storage-more-results-from-the-ndsa-storage-survey/

> The results of a survey given about current practices related to fixity checking. Provides an overview of what others are doing.

Fisher, Tim. *Checksum.* About.com: Computing and Technology, PC Support. http://pcsupport.about.com/od/termsc/g/checksum.htm

> Defines checksum, walks through a checksum scenario, and discusses available tools for calculating checksum values.

LeFurgy, Bill. *Hashing Out Digital Trust.* The Signal, Digital Preservation Blog. November 15, 2011. http://blogs.loc.gov/digitalpreservation/2011/11/hashing-out-digital-trust/

> A blog post that discusses checksums and other methods for implementing trust in digital environments.

Novak, Audrey. Fixity Checks: Checksums, Message Digests and Digital Signatures. ILTS Digital Preservation Committee. November 2006. http://www.library.yale.edu/iac/DPC/AN_DPC_FixityChecksFinal11.pdf

> Defines fixity, discusses their use in digital preservation, provides examples of practical implementations, discusses issues to consider, and discusses the best practices for Yale's University Libraries as of 2006.

PC Mag.com. *Checksum.* Encyclopedia. http://www.pcmag.com/encyclopedia_term/0,1237,t=checksum&i=39626,00.asp

> Computer based definition of checksum.

Riecks, David. The Trouble Transporting Tribbles (or File Verification using MD5 Checksums). Controlled Vocabulary. July 2010. http://www.controlledvocabulary.com/imagedatabases/file-verification.html

> Provides an overview of checksums and their uses and explores a few available checksum tools.

Vulnerability Issues

Daum, Magnus and Stefan Lucks. Hash Collisions (The Poisoned Message Attack) "The Story of Alice and her Boss". [2005]
http://th.informatik.uni-mannheim.de/people/lucks/HashCollisions/

This posting describes hashing and how and why someone might use hash collisions in a malicious manner.

United States Computer Emergency Readiness Team (US-CERT) website.
http://www.us-cert.gov/

Known checksum vulnerabilities can be researched through the United States Computer Emergency Readiness Team (US-CERT) website.

Selinger, Peter. *MD5 Collision Demo.* Published February 22, 2006, updated October 11, 2011.
http://www.mscs.dal.ca/~selinger/md5collision/

This post explains hash collision, describes collision examples, and why they might be harmful, and goes into technical details.

Stackoverflow.com
http://stackoverflow.com/questions/4581217/what-are-the-vulnerabilities-of-md5-and-how-can-they-be-remedied
http://stackoverflow.com/questions/1756004/can-two-different-strings-generate-the-same-md5-hash-code

Questions and discussions about the vulnerabilities of MD5.

WikiNews. *Chinese researchers crack major U.S. government algorithm used in digital signatures.* February 26, 2005.
http://en.wikinews.org/wiki/Chinese_researchers_crack_major_U.S._government_algorithm_used_in_digital_signatures

The vulnerabilities of SHA-1 are discussed.

Wikipedia. MD5 and SHA-1.
http://en.wikipedia.org/wiki/Md5 and http://en.wikipedia.org/wiki/SHA-1

General information on the hash algorithms that also discuss the vulnerabilities.