Hash-based File Content Identification Using Distributed Systems

York Yannikos[1], Jonathan Schlüßler[2], Martin Steinebach[1], Christian Winter[1], and Kalman Graffi[3]

[1] Fraunhofer Institute for Secure Information Technology
Darmstadt, Germany
{yannikos,steineba,winter}@sit.fhg.de

[2] University of Paderborn
Paderborn, Germany
sjonny@mail.upb.de

[3] University of Düsseldorf
Düsseldorf, Germany
graffi@cs.uni-duesseldorf.de

Abstract. A serious problem in digital forensics is handling very large amounts of data. Since forensic investigators often have to analyze several terabytes of data within a single case, efficient and effective tools for automatic data identification or filtering are very important. A commonly used data identification technique is using the cryptographic hash of a file and match it against white and black lists containing hashes of files with harmless or harmful/illegal content. However, such lists are never complete and miss the hashes of most existing files. Also, cryptographic hashes can be easily defeated e. g. when used to identify multimedia content.

In this work we analyze different distributed systems available in the Internet regarding their suitability to support the identification of file content. We present a framework which is able to support an automatic file content identification by searching for file hashes and collecting, aggregating, and presenting the search results. In our evaluation we were able to identify the content of about 26% of the files of a test set by using found file names which briefly describe the file content. Therefore, our framework can help to significantly reduce the workload of forensic investigators.

Keywords: Forensic Analysis Framework,File Content Identification,P2P Networks,Search Engines

1 Introduction

Currently one of the main challenges in digital forensics is an efficient and effective analysis of very large amounts of data. Today the investigation of just a single private desktop computer can easily result in several terabytes of data to be analyzed and searched for evidence or exonerating material. Widely used approaches to filter data utilize cryptographic hashes calculated from the file content and match these hashes against large white and black lists.

Since an important property of cryptographic hashes is collision resistance they are ideally suited for file content identification. The cryptographic hash of a file (subsequently also denoted as *file hash*) is solely calculated from the file content – meta data provided by the file system like a file name or a timestamp is not considered. The most commonly used file hashes in digital forensics are currently MD5 and SHA-1.

Filtering files by matching file hashes against white and black lists is both effective and efficient. However, it is impossible to create lists which contain the file hashes of all existing harmful or harmless files. Besides this, file hashes can be easily defeated especially when used for identifying multimedia files: For instance, just changing the format of a picture file makes its original file hash useless when trying to identify the modified picture file.

In this work we analyze what distributed systems throughout the Internet other than typical white or black lists are suitable to help identifying the content of files found within a forensic investigation. We propose a framework which can automatically search different suitable sources, e. g. search engines or P2P file sharing networks, for file hashes in order to find useful information about the corresponding files – such useful information are especially commonly used file names because they mostly describe the file content briefly. Our framework aggregates, ranks, and visually presents all search results in order to support forensic investigators in identifying file content without having to do a manual review. In our evaluation we give an overview on the effectiveness and efficiency of searching different distributed systems and on the popularity of different file hashes.

This work is organized as follows: In Sect. 2 a short overview about hash-based file content identification is given. Sect. 3 provides an analysis of different distributed systems with respect to whether or not searching file hashes is possible. In Sect. 4 we describe our file content identification framework and provide the evaluation of the framework in Sect. 5. Sect. 6 describes a typical use case for our framework, and in Sect. 7 we give a conclusion.

2 Hash-based File Content Identification

Today cryptographic hashes are used in forensic investigations mainly to automatically identify content indexed on a black or white list. The most commonly used cryptographic hashes in digital forensics are MD5 and SHA-1 which are supported by most forensic software providing mechanisms for hashing.

There has been only small work on utilizing different sources for hash-based file content identification besides large databases of indexed file hashes for black and white listing. One of the largest publicly available databases is the NIST National Software Reference Library (NSRL) which reportedly contains about 26 million file hashes usable for white listing [7].

Additionally, many databases for black listing exist. For instance, malware indexing databases like VirusTotal[4] or the Malware Hash Registry[5] are publicly available and can be queried for hashes of files containing malware. Other databases containing hashes of illegal material like child pornography also exist but are typically not available for the public. These databases are typically maintained by law enforcement agencies.

Other tools exist that are specialized in finding known files e. g. of cloud or P2P client software on a computer system. Examples are PeerLab[6] or FileMarshal [1] which are able to find installed P2P clients and provide information about shared files or the search history.

[4] `https://www.virustotal.com/`
[5] `http://www.team-cymru.org/Services/MHR`
[6] `http://www.kuiper.de/?id=59`

2.1 Advantages

Using cryptographic hashes for file content identification is very effective. Since cryptographic hashes are designed to be resistant to preimage and second-preimage attacks, i. e. regarding files it is very difficult to find the corresponding file for a given hash or to find two different files with the same hash, they are highly suitable to identify files. Additionally, calculating a cryptographic hash like MD5 or SHA-1 of a file is also efficient. For instance, to calculate the MD5 or SHA-1 hash of a 100 MB file can be done in less than 1 second on a standard computer system.

Another advantage of using cryptographic hashes is that besides the file content no other meta data regarding the file is required. This means that even a file that was modified in order to hide its illegal content, e. g. by renaming it using an obscure file name or file extension or by changing its timestamps, can be identified using its hash if it has already been indexed in a black list.

2.2 Limitations

Especially when used for searching known multimedia files, cryptographic hashes can be easily defeated due to its properties. For instance, just opening and saving a JPEG file will result in a new file which cannot be identified anymore by using the possibly black-listed cryptographic hash of the original file. Also, a criminal person sharing or distributing illegal material which is already black-listed may implement anti-forensic techniques to make an automatic identification of the illegal material extremely difficult.

As an example a criminal could easily implemented a web server module as content handler for pictures. This module could be designed to randomly choose and slightly change a certain number of pixels of each picture which is downloaded (e. g. to by viewed in a web browser). Compared to the original picture there would be no visible difference in the resulting picture. However, cryptographic hashing would completely fail in trying to identify the result as being basically the same as the original.

By using such a technique each criminal individual interested in downloading the illegal content would automatically get almost unique versions of the same content. Therefore, a criminal would not need to care about modifying the content on his own in order to hide it from forensic investigators. Such a scenario drastically shows that cryptographic hashing should not be used as a single mechanism to filter black- or white-listed data but rather as one of many different automatic approaches for file content identification.

Another approach commonly used for content identification not based on indexed material is the automatic detection of child pornography. This technique is typically implemented by applying a skin color detection in pictures or videos. However, skin color detection is not reliable in explicitly identifying child pornography since it produces very high false positive rates especially when considering *legal* pornography.

Using robust hashing for identifying child pornography based on indexed material, an efficient approach with low error rates was proposed in [10]. Unfortunately, robust hashing techniques are not yet used for file content indexing by P2P protocols, web search engines, and most hash databases, so we do not consider them in this work.

3 Distributed Systems for Hash-based File Search

In this section we analyze different distributed systems as potential sources regarding whether or not they are suitable for file hash searches. Therefore, we consider the popularity and technical properties of P2P file sharing networks, web search engines, and online databases of indexed file hashes.

3.1 P2P File Sharing Networks

Regarding to the popularity of P2P file sharing networks, Cisco reported in a recent study that in 2011 P2P file sharing networks were reason for about 15% of the total global IP traffic [3]. In another study the traffic in different regions was analyzed identifying P2P traffic as the largest share in each region, ranging from 42.5% in Northern Africa to almost 70% in

Eastern Europe [9]. The most popular P2P protocols were found to be BitTorrent, eDonkey, and Gnutella.

Although BitTorrent [4] is the most used P2P protocol with 150 million monthly active users [2], the protocol does not allow a hash-based file search. BitTorrent uses a *BitTorrent info hash* (BTIH) instead of the cryptographic hashes of the files to be shared. Since this BTIH is created from other data than just the file content, e. g. the file name, BitTorrent is not suitable for our purpose.

One of the most popular P2P protocols is the eDonkey protocol with 1.2 million concurrently online users[7] as of August 2012. The eDonkey protocol provides the technical properties for a hash-based file search since its *eDonkey hash* used for indexing is a specific application of the MD4 cryptographic hash [5]. The creation of the eDonkey hash is shown in Fig. 1. Basically, a file is divided into equally-sized blocks of 9500 KiB, each used to calculate the MD4 hash from. All resulting hashes are then concatenated to a single byte sequence where the final MD4 hash (the eDonkey hash) is calculated from.

An eDonkey peer typically sends his hash search request to one of the several existing eDonkey servers which replies with a list of search results. Besides this, eDonkey also allows a decentralized search using its *Kad* network, a specific implementation of the *Kademlia* distributed hash table (DHT) [6]. Further information on Kad can be found in [11].

Historically being the first completely decentralized P2P protocol, Gnutella uses SHA-1 for indexing shared files. Initially searching for SHA-1 hashes was supported but unfortunately has been disabled in most Gnutella clients due to performance reasons. A client still supporting SHA-1 hash searches is gtk-gnutella. However, using such a gnutella client yields only very few results since a SHA-1 hash search request is dropped as soon as it reaches a client not supporting the SHA-1 hash search. We also observed very long response times using the gtk-gnutella for SHA-1 hash searches in several test runs. Therefore, Gnutella is also not suitable for a hash-based file search due to effectiveness and efficiency reasons.

[7] Aggregated from `http://www.emule-mods.de/?servermet=show`

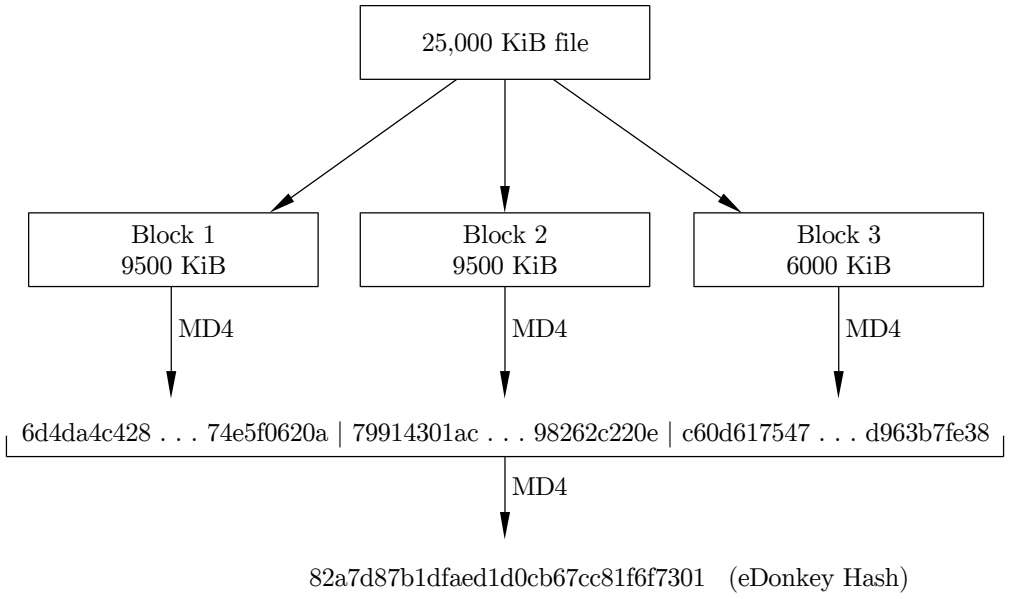


Fig. 1. Creation of an eDonkey hash

3.2 Web Search Engines

Since the success of web search engines is based on gathering as much Internet content as possible as well as applying powerful indexing mechanisms for fast queries on very large data sets, they provide a very good basis for a hash-based file search. Because many web sites like file hosting services also provide file hashes like MD5 or SHA-1 as checksums, the chances to find e. g. the name of a popular file when searching for its corresponding hash are relatively high. Fig. 2 shows a screenshot of a file hoster[8] providing useful information for a MD5 file hash.

According to [8] Google is by far the most popular search engine with about 84.4% market share. The follow-ups are Yahoo with 7.6% and Bing with 4.3%.

3.3 Online Hash Databases

Besides the National Software Reference Library (NSRL) of the NIST there exist many other hash databases to be searched online or to be downloaded and searched locally. For instance, the Malware Hash Registry[9] (MHR) by Team Cymru Research NFP contains hashes of files

[8] `http://d-h.st/`
[9] `http://www.team-cymru.org/Services/MHR`

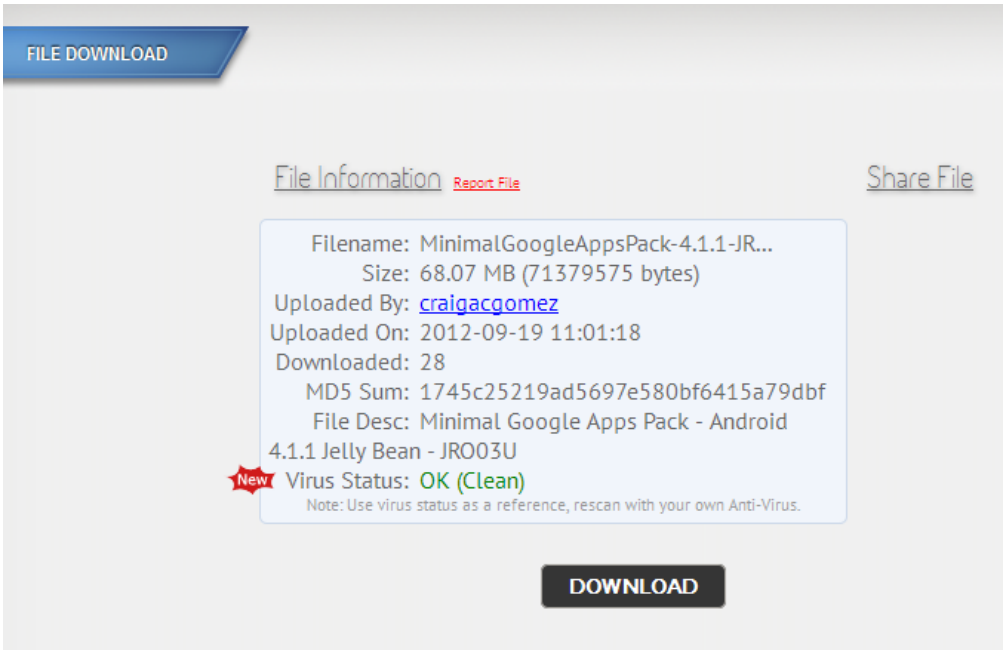


Fig. 2. Screenshot of a file hosting website providing useful information for a given MD5 file hash

identified as malware. The SANS Internet Storm Center Hash Database[10] (ISC) reportedly searches the NRSL as well as the ISC for hashes. Both the ISC and the MHR support search queries via the DNS protocol by forming a DNS TXT request using a specific file hash as subdomain of a given DNS zone. An example for querying the ISC database via the DNS protocol is given in List. 1.1. The result contains the indexed file name belonging to the hash ("Linux_DR.066") and the source ("NIST" – specifies the NSRL database).

```
> dig +short B47139415F735A98069ACE824A114399.md5.dshield.org TXT
"LINUX_DR.066 | NIST"
```

Listing 1.1. Querying the ISC hash database for a file hash using `dig`

4 File Content Identification Framework

When searching for a specific file hash, the most important information directly connected with the hash is a commonly used file name since it typically describes the file content briefly.

[10] `http://isc.sans.edu/tools/hashsearch.html`

In order to automatically search different sources and collect the results, we developed a modular framework prototype and implemented the following modules to search web search engines, P2P file sharing networks, and hash databases for file hashes:

- *eDonkey module:* Used to search eDonkey P2P networks for eDonkey hashes.
- *Google module:* Used to search Google for MD5 and SHA-1 hashes.
- *Yahoo module:* Used to search Yahoo for MD5 and SHA-1 hashes.
- *NSRL module:* Used to search the NIST National Software Reference Library for MD5 hashes. A copy of the NSRL was downloaded in advance to search the database locally.
- *ISC module:* Used to search the SANS Internet Storm Center Hash Database (ISC) for MD5 hashes.
- *MHR module:* Used to search the Malware Hash Registry (MHR) by Team Cymru Research NFP for MD5 hashes.

The file content identification process of our framework is shown in Fig. 3 and can be divided into the following steps:

1. Take a directory containing files as input
2. Calculate the MD5, SHA-1, and eDonkey Hash for each file within the directory and subdirectories
3. Search the mentioned search engines, P2P networks, and hash databases for these hashes
4. Collect all results, aggregate equal results
5. Present the results in a ranked list

5 Evaluation

To evaluate our framework we used a test set of 1473 different files with a total size of about 13 GB. Although being rather small the test set was designed to provide a good representation of files which are typically of high interest when found in forensic investigations. The following file categories were defined:

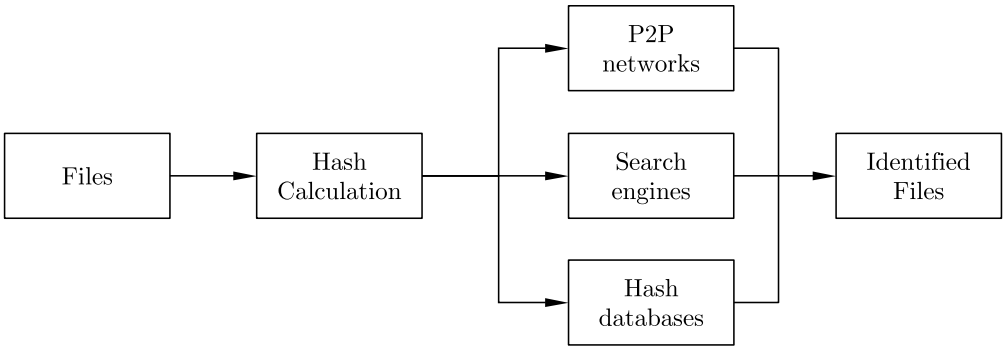


Fig. 3. Process for file content identification using cryptographic hashes

- *Music:* 650 audio files mainly containing music, randomly chosen from three large music libraries, various music download websites, or sample file sets.
- *Pictures:* More than 500 pictures mainly from several public and private picture collections, e. g. the 4chan image board.
- *Videos:* 34 video files, mainly YouTube videos, whole movies, or episodes of popular TV series.
- *Documents:* 240 documents in PDF, EPUB, or DOC (Microsoft Word) format, mainly from a private repository, including many e-books.
- *Misc:* Several archive files, Windows and Linux executables, malware, and other miscellaneous files, mainly taken from a private download repository.

5.1 Search Result Characteristics

Our framework collects meta data found by searching P2P file sharing networks, search engines, and hash databases. The search results are then presented in a ranked form based on how many different sources found equal meta data. This means that e. g. the more a specific file name is found for the hash of a file, the higher this file name is ranked. Together with the file name the framework also collects other meta data if available, e. g. URLs or timestamps.

In Table 1 we included the file names our framework was able to find for three sample files from our test set: one audio file, one picture, and one video file. As already mentioned,

locally available meta data, e. g. a local file name, is not considered by our framework since it cannot be trusted and may be forged in order to hide revealing information. The sample video file turned out to be a good example for the usefulness of the framework: Its local file name was `Pulp Fiction 1994.avi` thus a forensic investigator is likely to conclude that the file contains the well-known movie by Quentin Tarantino. However, in eDonkey networks our framework found several other more commonly used file names which were clearly describing the file content as actually being a different movie with rather pornographic content.

Like for the three sample files most file names found in our evaluation provided enough information to identify the corresponding file content.

Table 1. Sample results of a hash-based search using the framework to identify file content (local file name of the sample video file: `Pulp Fiction 1994.avi`)

Sample file	Source	Total number of results	Most common file names found (sorted by number of results in descending order)
Picture	Google	1100	`Lighthouse.jpg` `8969288f4245120e7c3870287cce0ff3.jpg`
	Yahoo	7	`Lighthouse.jpg`
	eDonkey	7	`Lighthouse.jpg` `sfondo hd (3).jpg`
	ISC	1	`Lighthouse.jpg`

Audio file	eDonkey	31	`Madonna feat. Nicki Minaj M.I.A. Give Me All Your Luvin.mp3`
			`Madonna - Give me all your lovin [ft.Nicki Minaj & M.I.A.].mp3`
			`Madonna ft. Nicki Minaj & M.I.A. - Give Me All Your Luvin'.mp3`

Video file	eDonkey	130	`La.Loca.De.la.Luna.Llena.Spanish.XXX.DVDRip.www.365(...).mpg`
			`La Loca De La Luna Llena (Cine Porno Xxx Anastasia Mayo(...).avi`
			`MARRANILLAS TORBE...avi`
			`Pelis porno - La Loca De La Luna Llena (Cine Porno Xxx(...).mpg`

5.2 Effectiveness

For 382 files in the test set (26%) we were able to find a commonly used file name which helped us to identify the file content. 220 of these files (almost 58%) yielded unique results, i. e. only a single source produced results for these files. Fig. 4 shows the total percentage

of files for which the individual sources successfully found a file name together with the aggregated numbers.

For Most results could be achieved searching eDonkey and Google. In comparison, searching Yahoo, ISC, or MHR produced no additional results. Besides Google and eDonkey the NSRL was the only other source that produced any unique results.

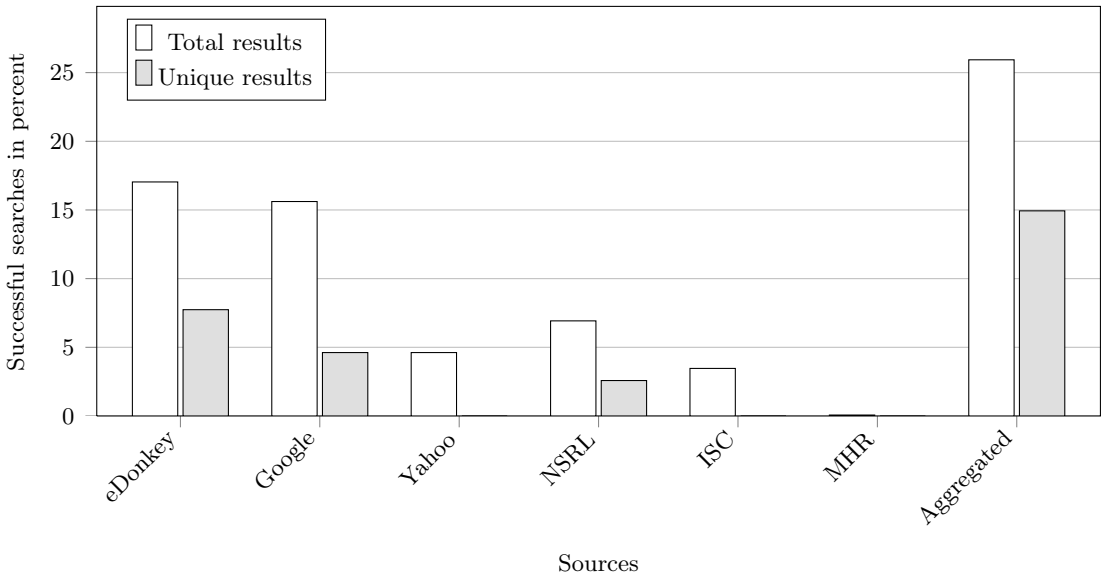


Fig. 4. Percentage of successful searches, for each source and aggregated

In Fig. 5 the search results for each category are shown. eDonkey turned out to be very effective in finding file names for given hashes of video files: It produced results for 53% of all video files of the test set. Additionally, eDonkey found a file name for the hashes of 18% of the pictures, 19% of the audio files, and 24% of the miscellaneous files.

Google was able to produce the most results for the miscellaneous files by finding file names for 66% of the hashes. The hash databases NSRL and ISC found the file names for most system files, sample wallpapers, or sample videos that are typically white-listed. Searching for documents like e-books or Microsoft Word files turned out to be very ineffective: Even by finding file names for only 3.8% of the document file hashes, Google was still the most successful source.

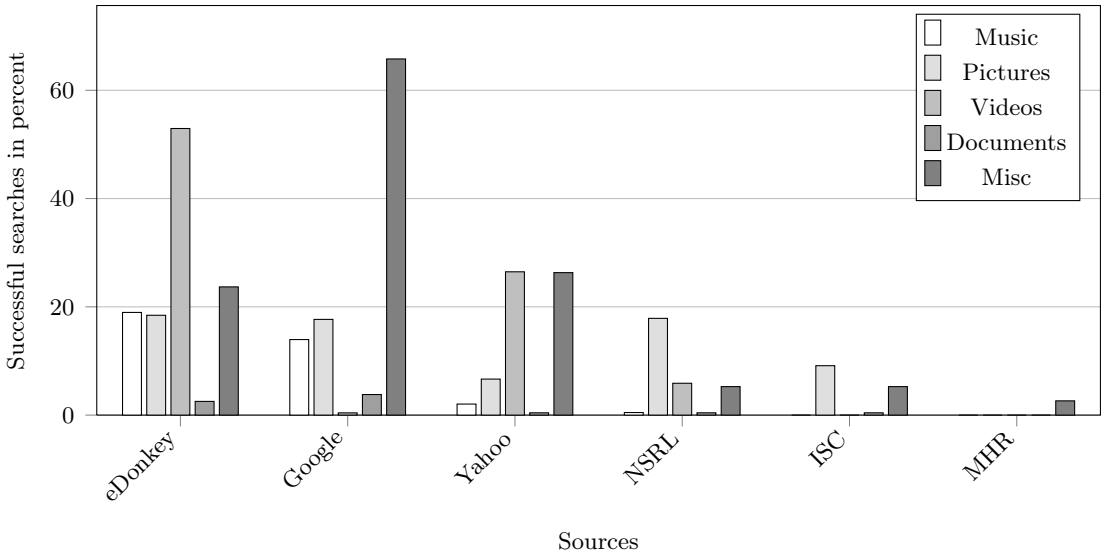


Fig. 5. Percentage of successful searches by category for each source

In addition to MD5 and SHA-1 hashes we also used Google to search for hashes of the SHA-2 family since they are also commonly used and supported by many forensic analysis tools. Our results are shown in Fig. 6. Using SHA-224, SHA-256, SHA-384, or SHA-256 hashes only yielded a small number of results and no unique results at all. Therefore, we think that using MD5 and SHA-1 can currently be considered sufficient for a reliable file content identification based on cryptographic hashes.

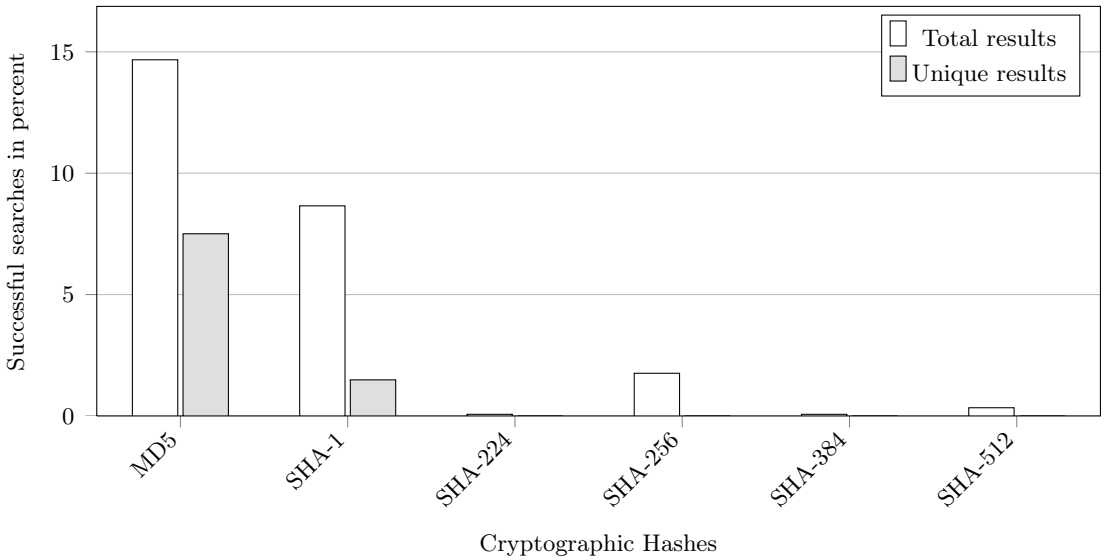


Fig. 6. Percentage of successful Google searches using different cryptographic hashes

5.3 Efficiency

To evaluate the efficiency of our framework we measured the average number of search requests per second that we were able to send to each source which can be seen in Fig. 7. We found that some sources could be searched very fast, e. g. the hash databases, and that others had implemented mechanisms like CAPTCHAs to throttle the number of requests within a specific time frame.

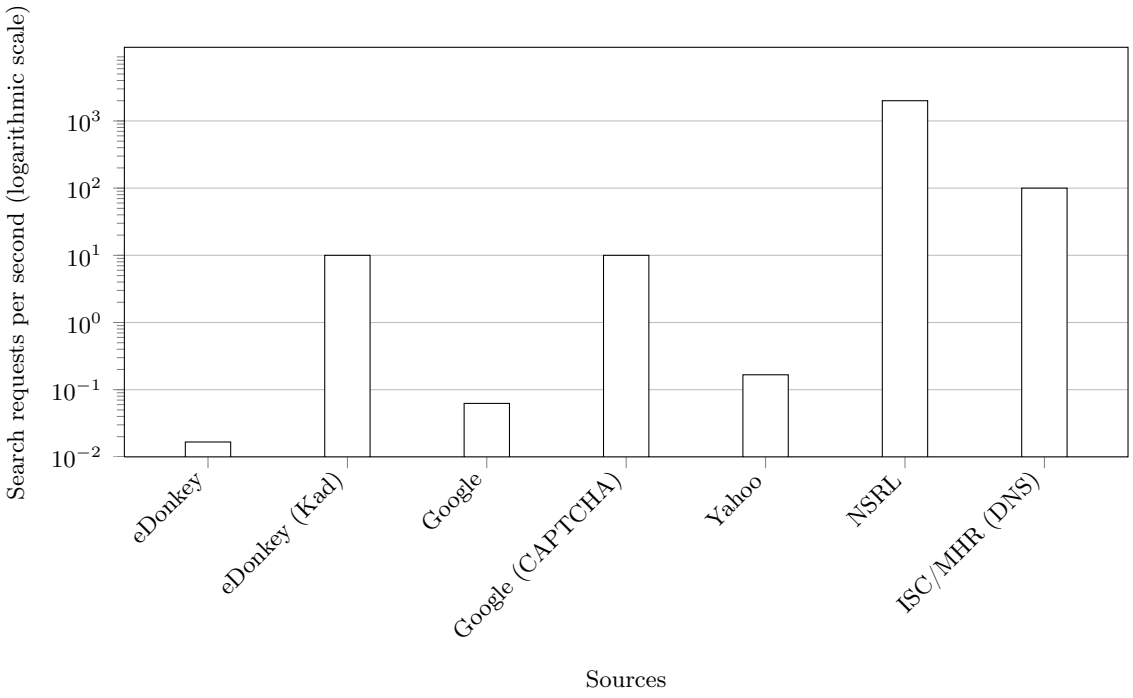


Fig. 7. Search requests per second for each source

eDonkey servers which are used for file search mostly have security mechanisms installed to prevent request flooding. Therefore, even if we sent just one request every 30 seconds, the servers quickly stopped replying to further requests. Eventually, we found that sending one request every 60 seconds reliably produced results without being blocked. However, using the decentralized Kad network of eDonkey (where peers are directly contacted with a search request) we were able to send requests much faster with at a rate of about 10 requests per second.

Unsurprisingly, the locally available NSRL hash database was the source which could handle most requests (about 2000) per second. Searching the ISC and MHR database was also very fast: We were able to send about 100 search requests per second using their DNS interface.

When using Google or Yahoo for searching, we found that the average rate for search requests was one request in 16 seconds for Google and one request in 6 seconds for Yahoo. Sending faster requests caused Yahoo to completely ban our IP with increasing ban durations up to 24 hours. In comparison, sending faster requests to Google resulted in getting CAPTCHAs to be solved.

Regarding Google, we observed that the search request rate affected the allowed number of search requests within a specific time frame before a CAPTCHA was shown. When sending requests faster than every 16 seconds, Google showed a CAPTCHAs after about 180 requests. When increasing the request rate to one request in every 0.2 to 2 seconds, Google already showed a CAPTCHA after about 75 requests. However, when we sent requests faster than every 0.2 seconds Google surprisingly allowed up to 1000 search requests until a CAPTCHA was shown. By adding an automatic CAPTCHA-solving service which require less than 20 seconds per CAPTCHA, we could theoretically increase the throughput of the Google search to about 10 search requests per second. However, since Google eventually started banning our IPs we have to do further research on this.

Fig. 8 depicts the number of allowed Google search requests before a CAPTCHA was shown in relation to the search request rate.

6 Use Case Description

A scenario which illustrates the usefulness of our framework is a typical forensic investigation where storage devices have been seized and are to be analyzed. After creating a forensically sound copy of the seized devices, an investigator typically uses forensic standard tools to do a first file system analysis. Additionally, he starts data recovery processes targeting unallocated

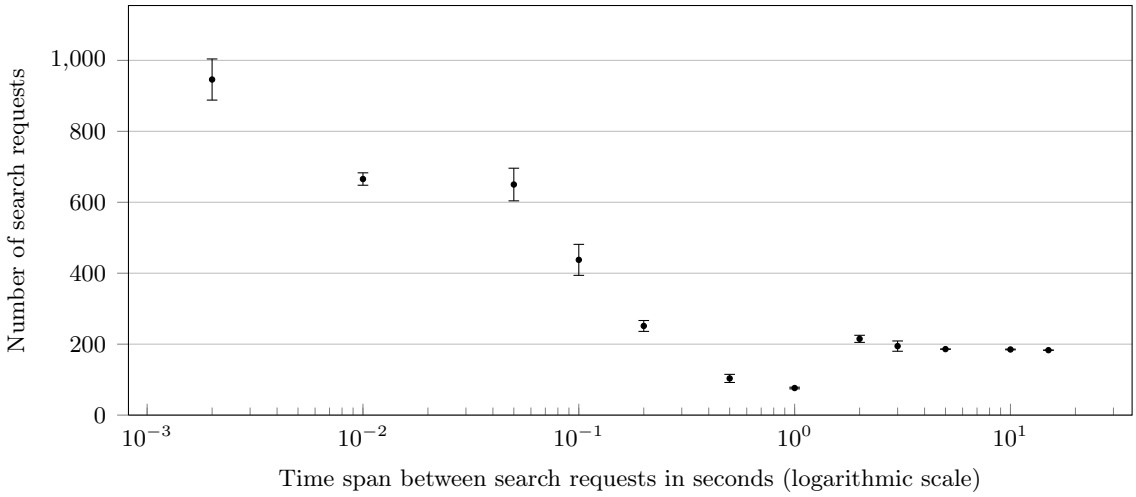


Fig. 8. Number of search requests (with error bars) sent to Google before a CAPTCHA was shown in relation to the time span between search requests

space to find previously deleted files that have not been overwritten. For recovered files a former file name is often missing especially when file carvers are used for recovery, e. g. because no corresponding file system information is available anymore. In the worst (but not untypical) case a forensic investigator ends up with a large amount of files with non-descriptive file names and almost no information about their content.

After such a first analysis the data has to reviewed in order to find incriminating or exonerating material. Common filter techniques based on white and black lists of cryptographic hashes are then used to reduce the amount of data to be reviewed as much as possible. This is where our framework supports the investigator's work significantly: Instead of only using filter approaches based on incomplete and probably deprecated hash databases the investigator can start our framework and feed all data to be reviewed into it. The framework automatically starts searching multiple available sources for any information about the data. Found white-listed files, e. g. system files or common application data, are automatically filtered. Black-listed files are marked as such and are presented to the investigator together with aggregated and ranked information about files which are neither white- nor black-listed but where information, e. g. a most common file name, has been found by searching P2P file sharing networks or search engines.

By going through the ranked list of search results, e. g. containing content-describing file names, a forensic investigator can find valuable information about the files and their content without having done a manual review so far. He can choose what files to sort out, e. g. because their content is not relevant for the investigation, or what files to review first, e. g. because of results that indicate incriminating material. Searching file hashes in distributed systems with large amounts of dynamically changing content provides a valuable addition to the currently used rather static white and black list-based filter approaches.

7 Conclusion

In this work we analyzed different distributed systems regarding whether they can help supporting file content identification in forensic investigations. We identified search engines and P2P file sharing networks as suitable systems for gathering information about files by searching for file hashes. To automate file content identification we implemented a framework to search the mentioned systems and to aggregate, rank, and present the search results. This supports forensic investigators in gaining knowledge about the content of files without requiring a manual review.

We implemented search functionality for the eDonkey P2P protocol as well as for the search engines Google and Yahoo. Additionally, we implemented interfaces to search the online hash databases NSRL, ISC, and MHR. In total, we were able to find useful information – mostly file names – for about 26% of the files in our test set (1473 files, total size 13 GB).

Searching eDonkey networks turned out to be very effective, especially for multimedia data like video files, audio files, and pictures. Google and the NSRL are also well suited for hash-based file identification. Besides this, Yahoo and the ISC and MHR hash databases did not produce any unique search results and thus can covered completely by using eDonkey, Google, and the NSRL.

Regarding efficiency, the hash databases were the fastest to search, followed by eDonkey's Kad network and most probably Google in combination with an automatic CAPTCHA solver.

We found that MD5 and SHA-1 are still the most used file hashes and therefore are well suited for a hash-based file search. Other hashes, e. g. of the SHA-2 family, are only rarely used within the Web.

We suggest further research on identifying other suitable distributed systems for hash-based file search, e. g. other P2P file sharing networks like DirectConnect. Additional databases providing specific multimedia content like Flickr[11], Grooveshark[12], or the IMDb[13] could also be used to collect additional information about file content, e. g. in order to verify or extend search results.

Other improvements can be done regarding the framework architecture by introducing a client–server model for parallelizing search tasks: A network of multiple file hash search servers could be used to process search tasks created by individual clients controlled by forensic investigators. A cloud service infrastructure may also provide possibilities for faster distributed file hash searches.

Acknowledgment This work was supported by CASED (www.cased.de).

References

1. Adelstein, F., Joyce, R.: File Marshal: Automatic extraction of peer-to-peer data. Digital Investigation 4, 43–48 (2007)
2. BitTorrent, Inc.: BitTorrent and μTorrent Software Surpass 150 Million User Milestone; Announce New Consumer Electronics Partnerships (2012), http://www.bittorrent.com/intl/es/company/about/ces_2012_150m_users
3. Cisco Systems, Inc.: Cisco visual networking index: Forecast and methodology, 2011–2016 (May 2012), http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf

[11] http://www.flickr.com/
[12] http://www.grooveshark.com
[13] http://www.imdb.com/

4. Cohen, B.: Incentives build robustness in BitTorrent. In: Workshop on Economics of Peer-to-Peer systems. vol. 6, pp. 68–72 (2003)
5. Kulbak, Y., Bickson, D.: The eMule protocol specification. School of Computer Science and Engineering, The Hebrew University of Jerusalem, Tech. Rep (2005)
6. Maymounkov, P., Mazieres, D.: Kademlia: A peer-to-peer information system based on the xor metric. Peer-to-Peer Systems pp. 53–65 (2002)
7. National Institute of Standards and Technology: National Software Reference Library (2012), http://www.nsrl.nist.gov/
8. Net Applications: Desktop Search Engine Market Share (October 2012), http://www.netmarketshare.com/search-engine-market-share.aspx?qprid=4&qpcustomd=0
9. Schulze, H., Mochalski, K.: Internet Study 2008/2009 (2009), http://www.ipoque.com/sites/default/files/mediafiles/documents/internet-study-2008-2009.pdf
10. Steinebach, M., Liu, H., Yannikos, Y.: Forbild: efficient robust image hashing. In: Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series. vol. 8303, p. 18 (2012)
11. Steiner, M., En-Najjary, T., Biersack, E.: A global view of Kad. In: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement. pp. 117–122. ACM (2007)