



Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin

Digital forensics as a service: Game on

H.M.A. van Beek*, E.J. van Eijk, R.B. van Baar, M. Ugen, J.N.C. Bodde, A.J. Siemelink

Netherlands Forensic Institute, Laan van Ypenburg 6, 2497 GB The Hague, The Netherlands

ARTICLE INFO

Article history:

Received 12 March 2015

Received in revised form 8 July 2015

Accepted 11 July 2015

Available online xxxx

Keywords:

Distributed systems

Digital forensics

Big data

XIRAF

HANSKEN

ABSTRACT

The big data era has a high impact on forensic data analysis. Work is done in speeding up the processing of large amounts of data and enriching this processing with new techniques. Doing forensics calls for specific design considerations, since the processed data is incredibly sensitive. In this paper we explore the impact of forensic drivers and major design principles like security, privacy and transparency on the design and implementation of a centralized digital forensics service.

© 2015 Elsevier Ltd. All rights reserved.

Introduction

Many papers in the field of digital forensics start with the observation that the size of digital material increases, that the complexity and diversity of the digital evidence grows and that more advanced techniques are needed to be able to keep up with the evolving digital society.¹

Since December 2010, the Netherlands Forensic Institute has been using a service-based approach for processing and investigating high volumes of seized digital material: Digital Forensics as a Service (DFaaS) (van Baar et al., 2014). This service is called XIRAF (Bhoedjang et al., 2012).

Now, four years later, this approach has become a standard for hundreds of criminal cases and over a thousand investigators, both in The Netherlands and abroad. After having processed over a petabyte of data, we have experienced the impact of the XIRAF system and the paradigm shift it is causing (van Baar et al., 2014). XIRAF started in 2006 as a scientific research project aimed at identifying

and developing techniques for automating (parts of) the data analysis process. XIRAF was never meant to be an operational system for processing petabytes of data and providing access to over a thousand investigators. As a result, design decisions taken during the development of XIRAF leave room for improvement.

In the beginning of 2012, we started working on the successor of XIRAF, named HANSKEN. This work consisted of defining design principles, building a proof of concept (PoC) based on the new principles and ideas, making design decisions based on the principles and PoC and building a production version to replace XIRAF. This paper provides an overview of the major design decisions that form the foundation for the HANSKEN solution for providing digital forensics as a service.

A lot of challenges arise when building a system to provide insight in petabytes of different types of data. Especially when integrity and confidentiality of the data are crucial. While it is tempting to focus on developing new techniques and building bigger and faster systems, boundaries need to be established in which such a system can operate. Without these boundaries, major risks of data breaches and leaks of sensitive information exist.

* Corresponding author.

E-mail address: harm.van.beek@nfi.minvenj.nl (H.M.A. van Beek).¹ <http://apmdigest.com/gartner-top-10-strategic-technology-trends-for-2013-big-data-cloud-analytics-and-mobile>, visited March 11, 2015.

In Section 2, the reasons why a forensic big data solution is desirable are described (forensic drivers), as well as the motivation for the boundaries of such a system (design principles). Section 3 contains considerations for how the forensic drivers and design principles affect the chosen solutions. Section 4 describes different solutions we have implemented in HANSKEN in order to cope with the considerations while still being able to do digital forensics. A lot of work has been done in the field of forensics and big data, even though the term was not yet used in most related work. Section 5 discusses different topics related to big data and how we see them match with a forensic big data platform like HANSKEN. Finally, we draw conclusions in Section 6.

Motivation

Business needs provide the main reasons for developing and providing a centralized system for doing large scale forensic data analysis. First of all, cost reduction asks for automating parts of the extraction and analysis process. Here, the economies of scale apply (Armbrust et al., 2010). Secondly, centralization makes it possible to standardize forensic data extraction and analysis and increase its quality. All this is explained in detail below.

As mentioned, the Netherlands Forensic Institute has been providing Digital Forensics as a Service to the Dutch law enforcement organizations since December 2010. Fig. 1 shows the procedure of handling digital forensic cases using this approach.

On the right, there are detectives and analysts that have questions related to information presumably available in the digital material shown on the left. To guarantee forensic integrity, forensic images are needed (van Baar et al., 2014; Kohn et al., 2013), so the first task is to create these forensic copies of the digital devices. The images are copied to a central storage and processed using a standard set of tools. We call this the *extraction process*. The applied tools range from tools that analyze file systems, extract files, carve unallocated space and create full text indexes, to tools that parse chat logs, browser history and e-mail databases. The results of these tools, i.e. the extracted *metadata*, are stored in a centralized database. The combined data and metadata of this process are referred to as *traces*, e.g. an e-mail, chat message or zip archive. After storing these traces, they can be queried using multiple methods: detectives can log on using a web browser and query the traces by applying filters and text searches. Digital investigators can use the programming interface to run automated tools and scripts written in their favorite programming language. Analysts may want to retrieve all information and analyze the results using data visualization tools, integrate additional data sources or build a network of contacts, for example. This makes it possible to identify, classify, organize and compare the traces within seconds, based on hypotheses and questions the investigators have. This can be done at any time during the investigation.

To support this process, the next paragraphs discuss the three drivers, eight design principles and our two ways of looking at data in the system. This defines the scope within which we designed HANSKEN.

Forensic drivers

Our main goal is to provide a service that processes high volumes of digital material in a forensic context and gives easy and secure access to the processed results. We identify three main forensic drivers: minimization of the case lead time, maximization of the trace coverage and specialization of people involved. These forensic drivers are the reasons for building a big data forensic platform.

Minimize case lead time

Generally, the first 48 hours of an investigation are critical to an investigation (U.S. Department of Justice, Office of Justice Programs, Office of Juvenile Justice and Delinquency Prevention, May 1998; Joyce, 2012; Wikipedia). Traditionally, results from digital investigations are not available in these first days. Traces found in digital material are therefore often used for validating hypotheses instead of forming them. In an ever increasing digital society, digital evidence becomes more and more key evidence. This makes it unacceptable to exclude digital material from the initial response: digital material must be available to the investigation team in those 48 hours. In this context, available does not only mean that investigators have access to the data, but also that they have tooling at their disposal for finding relevant traces. Examples are high performance filtering tools based on trace details or keywords and visualization tools for presenting search results.

To give access to the traces within 48 hours, processing of the seized material must be automated. This has high impact on the way digital material should be handled (van Baar et al., 2014). Furthermore, the results of this automated process must be made available to the investigation team directly and not to specialized digital investigators. This is discussed below. Since investigation teams can be scattered over multiple locations, access to the data and extracted traces should not be limited by e.g. building or department boundaries.

To speed up the investigation, detectives should be able to annotate or tag interesting traces or traces they do not understand. Other detectives and digital investigators must have access to the annotation so that they can act on it.

Maximize coverage

Seized material varies wildly, both in types of devices (hard drive, volatile memory or mobile phones), but also in file systems and file formats contained in forensic copies made from these devices. This is caused by simple things like software upgrades and the availability of new devices and new software for existing devices. This variation requires constant attention to make sure that the traces contained in the data are extracted. To keep up with software upgrades as well as counteract the ever increasing sophistication in technology used by suspects, increasing sophistication in the trace extraction tools is needed.

Processing petabytes of data in a central environment means processing a large variety of images. This requires the tools to process many different file formats, database formats and applications. Centralized processing of more and more data results in increased insight in the coverage

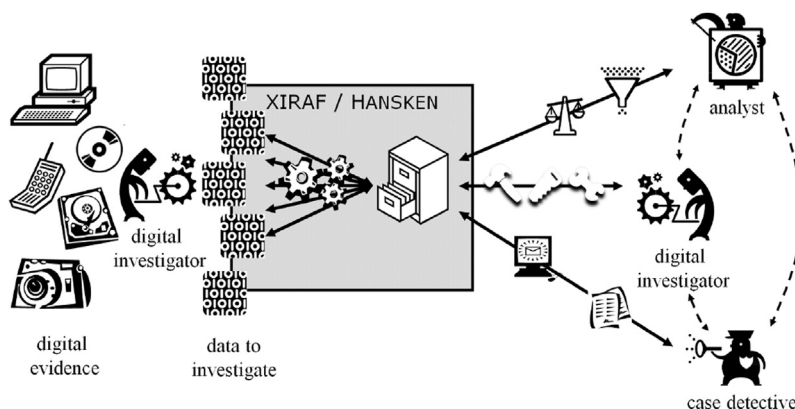


Fig. 1. : Digital forensics as a service.

of seized material. Analysis of the processing results makes it possible to constantly observe and increase the quality of the extraction process.

Freeing up digital investigators to perform more in-depth research will fortify this upward spiral, maximizing the coverage. When more investigative methods are developed, turnaround time and time spent for digital investigations will reduce. By embedding newly gathered knowledge in the service, the system is used as a knowledge center. New knowledge gained by research directly helps other investigative teams that encounter the same problems.

Finally, in the dynamic environment where new devices and software is released daily, setting up an environment that supports *continuous delivery* (Humble and Farley, 2010) is needed to keep the service up to date.

Specialize people

Case detectives should be the ones looking at the digital material. They can use their valuable case knowledge for identifying *relevant* traces. Therefore, they must be provided with tools to query the traces, filtering out irrelevant ones, preferably within seconds. When relevant traces are found, they must have direct access to the source material, like pictures, document and e-mails.

It is true that detectives run into problems understanding certain concepts, like carved files or the interpretation of a time stamp, but they can be educated. As described by Bhoedjang et al. (Bhoedjang et al., 2012), the benefits of non-experts investigating digital material outweigh the risks of misinterpretation.

Digital investigators help detectives by explaining the query results and do more in-depth investigation on technical details of *relevant* traces. This collaboration between detectives and digital investigators is crucial in understanding digital evidence. Traditional procedures make it hard to collaborate. Digital investigators report large result sets that are scattered across multiple detectives. Just like with a printed dossier, splitting up the evidence makes it harder to keep a bird's-eye view. In general, not material but investigative topics are distributed among detectives. The service model makes it possible to analyze all data in

the context of a topic. A trace that is irrelevant to one topic, might be decisive to another topic in the same case.

To best support detectives during the investigation, digital investigators should *not* be tasked with system administrative tasks that are better performed by dedicated system administrators. In most situations, digital investigators are held responsible for all aspects of their investigation environment (storage, network, software, security, etc.). Combined with their central role in securing and analyzing digital evidence, this leads to a lot of administrative overhead, possible security breaches, failing backup systems and the use of obsolete software, amongst others. Digital investigators are found to be either under-qualified or overqualified for a lot of the tasks they perform on a daily basis. In the DFaaS setup in The Netherlands, digital investigators focus on the forensic tasks, i.e. seizing material, extracting data from it and do in-depth research on selected traces. The data is sent to a centralized system that automatically extracts traces from the data and gives digital investigators, detectives and analysts access to the traces. Several administrators execute domain-specific tasks related to this service, like application administration, database administration, storage administration and infrastructure administration.

Design principles

Although the XIRAF system was never meant to be put in production in its current scale, providing digital forensics as a service using XIRAF has taught us a lot. The changing roles of digital investigators, detectives and analysts are set out above. One of the most important lessons learned is that even though it is tempting to constantly add new and shiny features and functionality, the foundations of the system need to be thoroughly thought out. This reduces the risks that come with putting this much sensitive data in a central location, sharing the infrastructure with different departments or even organizations. Investing in a large centralized system helps us to realize the forensic drivers, but to make such a big investment viable, it requires a large scale deployment.

Providing access to seized material via a centralized service makes the investigation team depend on a remote

system and remote support team. Where in a traditional investigation the digital investigators administered their own hard- and software, they do not rely on system administrators that may not be part of their own organization and on hardware that is shared with other departments.

The risks of a big data platform and other considerations have led us to define eight principles that we held to while designing and implementing HANSKEN. In order of priority they are:

1. Security
2. Privacy
3. Transparency
4. Multi tenancy
5. Future proof
6. Data retention
7. Reliability
8. High availability

With priority we mean that if a solution implemented for principle 6 (data retention) clashes with a higher principle, for example 3 (transparency), the solution is not implemented and a different solution needs to be chosen. The three most important principles are driven by sociological needs, the other ones are mainly driven by business needs. The principles apply to all aspects of the system: the forensic data, the way this data is used in an investigation, and the hardware, network and software design.

Security, privacy and transparency: forensic data and forensic professionals

The security principle encompasses all three corners of the CIA triad (ISO, 2013): confidentiality, integrity and availability. In the forensic context, integrity and authenticity (the fourth corner to the CIA “triad” that is always added) (Tipton, 2009) are paramount, which is reflected in security being our number one principle. The explicit principles *reliability* and *high availability* were added since they highly impact the design of a distributed implementation.

In general, material is seized for a specific case with a specific suspect and available exclusively to a limited team of investigators for a limited amount of time. Seized material contains a lot of personally identifiable information (PII), like identities of suspects, victims, but probably also their relatives. Although this data is available to the investigation team, this does not mean that others should have access to it. Furthermore, it may not be required for all team members to have access to all details of all material. The privacy of everyone involved with a crime must be protected. The sociological principles dictate us that the data should be made available to the investigators only (data security), that the privacy of the identities in the seized material has to be taken into account (data privacy) and it must be possible to review how certain traces were created and accessed (data transparency).

The three sociological principles apply to the digital investigation process as well. Administrators, for example, need insight in the load, i.e. the number of

concurrent users, queries, et cetera, to keep the service up and running. They do not need to know details on who is accessing which case and what he or she is querying for. So, only authorized people are allowed to perform certain operations (professional security), the privacy of the users of the service needs to be protected (professional privacy), but it must be possible to trace who did what with the seized material (professional transparency).

Business principles

Next to the three sociological principles, we identify five business principles that define the boundaries of the system. They form the base for the requirements on the setup and organization of the (technical) environment.

Multi tenancy

Since we want to service multiple users, probably even multiple organizations (tenants), it is required that no information is disclosed between tenants. Moreover, the impact of many users and complex queries should be minimal on the other tenants.

Future proof

Development of HANSKEN started in 2012. After three years of discussion, design and implementation, we have built a system where we feel confident we can keep up with growing data for the next years. New insights or techniques may however warrant replacement of parts of the system. This shouldn't result in a complete redesign of the system or that parts of the system function differently due to a new implementation.

Data retention

A lot of data will be processed by HANSKEN. If the service is the only place where forensic data is stored, we need to make sure this data is not lost due to system failure or user errors. This is true for both the images created from seized material, but also for logs generated by the use of the system. The main driver for this principle is legal obligation.

Reliability

When working with forensic data, the results should be predictable: the investigator must be able to *rely* on the extraction and analysis results. This means that when we perform the extraction or query the data, the output is deterministic and not dependent on factors outside the forensic process, e.g. load on the cluster, disks failing or a large number of concurrent users. Of course, the data itself must be reliable too. This so-called data integrity is covered in our security principle as mentioned before.

High availability

As explained in the introduction, digital forensics is an ever increasing field and the amount of evidence found in digital material continues to grow. This means that investigators continuously need access to the digital material, preferably 24/7.

Design considerations

In this section we motivate our most important design considerations. While the three forensic drivers push towards more functionality and unrestricted access of the data, e.g. a developer looking at the case data to develop new forensic methods, our eight principles push back to reduce impact of a big data platform on security and privacy of suspects, victims and users of the system. Most considerations are heavily related to a forensic big data service solution, a violation has less impact for a single machine solution with a single investigator analyzing a small data set. The described considerations touch upon different parts of the motivation and even though the forensic drivers and design principles may work in opposite direction, we have found some solutions that help both sides. Table 1 shows which consideration touches upon which motivation.

With respect to security and privacy, we follow the ideas of the Jericho Forum (Lacey, 2005). Their main strategy is to be able to operate in a de-perimeterized environment (Forum, Dec. 2006), meaning the protection of systems and data on multiple levels by using a mixture of encryption, inherently secure protocols, inherently secure computer systems and data-level authentication rather than the reliance of an organization on its (network) boundary/perimeter to assumed hostile external systems and networks. With respect to security, the Jericho strategy states that data should protect itself. In our context, it must also prove itself to be forensically sound (data integrity). With respect to privacy, the Jericho strategy is to have multiple identities, derived from one core identity (Forum, May 2011). This implies security and privacy by design.

With respect to cryptography, we follow Kerckhoffs's principle (Petitcolas et al., 2011), stating that the system should be secure even if everything about the system, except the key, is public knowledge.

The system is designed with a 'trust no-one' philosophy. Although data is seized by detectives and handed over to digital investigators for imaging, this does not mean that all people involved in processing the data should have access to it. For example, the developers and system administrators (who are not part of the investigation team in the service model) have no business need to look inside the data. This means that the data has to protect itself.

In the remainder of this section, we discuss the considerations, summed up in Table 1.

Reusability of forensic knowledge

The forensic knowledge about file system formats, file formats and other forensic knowledge obtained through specifications, reverse engineering and publications should not be stored in HANSKEN. We externalize this knowledge to libraries that offer a standardized way (API) to export this knowledge. Furthermore, we define a data model at a level similar to this API, to store data and traces in XIRAF and HANSKEN. The data model and libraries are already used in XIRAF, so they are battle-tested with over a petabyte of data.

The advantage of using libraries is that they can be used in other tools and have their own maintenance cycle. Since

Table 1
Relation between design considerations (rows) and forensic drivers and design principles (columns).

	Maximize coverage	Minimize lead time	Specialize people	Data security	Data privacy	Data transparency	Professional security	Professional privacy	Professional transparency	Multi tenancy	Future proof	Data retention	Reliability	High availability
1. Reusability of forensic knowledge	x		x								x		x	
2. Distributed extraction		x												x
3. Image format		x	x	x						x	x	x		
4. Key management			x	x										
5. Encrypted communication			x	x	x		x							
6. Unique identifiers for images			x											
7. User management			x	x			x			x				
8. Logging framework			x					x						
9. Transformation of confidential information			x		x			x						
10. Tool responsibility	x		x										x	
11. Business rules	x		x							x				
12. Best coding practices		x									x			x

the amount of forensic knowledge in HANSKEN is reduced to a minimum, the system can be built and maintained by developers without a forensic background. Maintaining the libraries is the responsibility of forensic investigators. If they add new forensic knowledge, e.g. a new e-mail format or a new property of an existing file system, this is automatically incorporated into the extraction process.

Distributed extraction

Bringing the tools to the data

XIRAF makes it possible to apply tools to a forensic image on a single machine. As a result, this *extraction process* does not scale. Furthermore, this scheduling process is iterative, where in each iteration, all tools check if there is data that they can process. If so, the tool processes the data, resulting in new traces or more details on the existing trace. For example, initially, the volume tool extracts volumes from a disk, the next iteration the file system tool identifies file systems and extracts files from them. Once files are available, lots of tools can be applied to them, like the hashing tool, mime typing tool, e-mail tool and chat tool. All these tools are applied subsequently. This implies that the data needs to be extracted from the image multiple times: we bring the data to each tool. Results are stored in an intermediate file, until the extraction is complete. After having processed all traces, the results of the extraction process are published to a database to make traces available for querying. This publication step is a manual process within XIRAF.

To make it possible to process one forensic image using multiple machines, we use distributed technology in HANSKEN. Apart from that, we apply either all tools to a trace, or no tool at all. The size of tools is generally relatively small compared to the size of data that they need to process. Thus instead of bringing the data to the processing tools, we bring the tools to the data. This means that as soon as we have the data available, we apply all tools to it. As soon as the data is read from the image, it is kept in memory and all tools are applied. Once a trace is fully processed, the results are stored in a database so it can be queried while other traces are still being extracted. This means that the first traces are queryable minutes after the analysis starts.

Data driven acquisition

Preferably, we start the process of extracting traces from a forensic image as soon as the first bits of a device are uploaded to the central system. In order to extract traces from an NTFS file system, for example, the master file table (MFT) must be extracted first. The MFT is generally not stored at the beginning of the disk. When the extraction process can request the blocks containing the MFT to be served first, it can start extracting the file meta data, thus reducing the time to deliver these to the investigating team. Furthermore, depending on the type of case (like financial fraud or child pornography) some file might be more valuable than others. After examining the MFT, the blocks for case-specific files can be requested subsequently.

Serving the blocks containing the MFT first, makes it possible to start analyzing it. Therefore, we designed an image format (see below in Section 3.3) that splits the

image data in encrypted blocks. The image format supports unordered blocks, that is the blocks in the image do not necessarily need to follow the order of the original data. This makes it possible to implement dynamic pipelining: the extraction process influences the imaging process by asking for certain blocks of data to become available with priority, like the block containing the MFT. This may lengthen the time it takes to image a disk due to the random accesses, but the total processing time of the imaging plus extraction process is shortened.

We adopted this concept of data driven acquisition in the design of HANSKEN. This does not necessarily mean that the acquisition must be combined with the uploading of the data to the central system. HANSKEN also supports the conversion of an earlier acquired image to the new format. Commonly used formats are supported, like raw and the EnCase evidence image file format. The target location of the newly created image can be a local file system, but also the central system, resulting in a direct upload.

Image format

We designed our own (forensic) image format. The main need for this format is that we want to compress and encrypt the seized data, though we have to be able to randomly access the data by multiple tools in parallel in a distributed environment. Another need is described in the previous section: We want to be able to read and store data in a different order than starting at the first byte and ending at the last. To still be able to efficiently search and process the data, we add an index file containing the location of individual blocks. This index file also contains calculated hashes for validating data integrity and authenticity. To store details about the imaging process like information about the acquisition, we add an optional third file. The names of these three files correspond, though the latter two get an additional extension.

To protect the data, it is compressed and encrypted as soon as it is read from the seized material. The image format supports multiple compression and encryption mechanisms. Initially, the cryptographic key to decrypt the data is only available to the investigator who set up the imaging process. He is trusted with the physical device and thus has access to the data. So security is taken into account directly from the point that the data enters the system. To make sure that the key and the image are kept apart, the key must be stored in a different security domain from the encrypted image.

It might be possible to use the Advanced Forensic Framework 4 (AFF4) (Cohen et al., 2009) file format, though our specific requirements on the combination of performance, encryption, compression, random block order and random access makes a new implementation preferable and justifiable.

In-depth details on the design and implementation of the image format are beyond the scope of this paper.

Key management

We use encryption for securing the data processed in HANSKEN. As mentioned, the service itself only contains the

encrypted data, the encryption keys are stored outside the service, in a separate security domain. We prefer a solution where the images are stored in a centralized system and the cryptographic keys are kept at the department where the physical material resides. So, to access data inside the service, the requesting party has to provide information needed for decrypting the data. This applies both to users who want to delegate the extraction process to the service as well as to users who want to query the extracted traces.

Key storage

To be able to securely store the keys, we designed an algorithm that makes sure the following conditions are met:

- all data is encrypted using a key;
- all keys are stored in a remote key store;
- all data in the key store is encrypted based on a public/private key pair;
- the cryptographic key to the data itself is not stored directly;
- authorized users have access to a shared secret only (which is *not* the key);
- only the service can reveal the key using this shared secret;
- neither the key nor the shared secret has to be communicated when accessing the service;
- only users who have been granted access to the shared secret (initially the uploader of the image) can grant others access to the data by sharing this secret.

A prerequisite for the algorithm is that the central service as well as all users involved with the extraction and analysis must have a public/private key pair. We define functions for encrypting with either the private key (sign) or public key of a user:

$(p, q)_u$: public/private key pair for user u
 $E_k(b)$: encrypt b using key k
 $PUB_u(b) = E_p(b)$ for key pair $(p, q)_u$
 $SIGN_u(b) = PVT_u(b) = E_q(b)$ for key pair $(p, q)_u$
 $b = PUB_u(PVT_u(b)) = PVT_u(PUB_u(b))$

When storing data for image i in the central system, we start by generating two keys: encryption key k_i for encrypting the data and shared secret s_i for obfuscating the key. The shared secret is available to the user and must be provided to access the data. The “key” can be calculated by the HANSKEN service if and only if the shared secret is provided.

k_i : encryption key for image i
 s_i : shared secret for image i

For the chain of custody, we want to validate that the uploader is the person that actually generated the shared secret. This is done by signing the shared secret using the private key of the uploader (u):

$$s'_i = SIGN_u(s_i)$$

Key k_i , used for encrypting image i , and the signed shared secret s'_i are not stored directly. We obfuscate (using a bitwise exclusive or operation) key k_i using signed shared secret s'_i . This breaks the key in two. Next, we encrypt the obfuscated key using the public key of the HANSKEN service that must be able to access the (unencrypted) data:

\oplus : bitwise exclusive or operator
 $PUB_u(s'_i)$: user part of the key
 $PUB_S(k_i \oplus s'_i)$: service part of the key

The keys can be stored in any (publicly available) database: a key store. If a user u wants to access data of image i , he needs to retrieve encrypted shared secret $PUB_u(s'_i)$ from the key store, decrypt it using his private key

$$PVT_u(PUB_u(s'_i)) = s'_i$$

and provide it to service S . This service retrieves the obfuscated key $PUB_S(k_i \oplus s'_i)$ from the key store, decrypts it

$$PVT_S(PUB_S(k_i \oplus s'_i)) = k_i \oplus s'_i$$

and resolves k_i using the provided shared secret:

$$(k_i \oplus s'_i) \oplus s'_i = k_i$$

To grant another person v access to the data of image i , the uploader u (or any other who has access to the shared secret) needs to decrypt s'_i using his private key

$$PVT_u(PUB_u(s'_i)) = s'_i$$

and encrypt it using the public key of person v that requires access:

$$PUB_v(s'_i) = PUB_v(PVT_u(PUB_u(s'_i)))$$

Now, user v can access the data too by retrieving the encrypted shared secret, decrypt it using his private key and provide it to the service.

Key exchange

The solution described in the previous section does not meet the condition that neither the key nor shared secret is communicated when accessing the service.

The obfuscated encrypted obfuscated key $PUB_S(k_i \oplus s'_i)$ can be communicated, since it is encrypted with the public key of the service. However, signed shared secret s'_i needs to be decrypted by the user wanting to access the data. To make sure that this shared secret is also encrypted in transport, a session between a user and the service starts with negotiating a session key t , based on Diffie-Hellman key exchange (Diffie and Hellman, 2006). Basically, both sides generate a random number: t_u for the user and t_s for the service:

t_u : random number for user u
 t_s : random number for service S

These random numbers are encrypted with the public key of the communicating party ($PUB_u(t_s)$ and $PUB_S(t_u)$) and exchanged. The communicating party decrypts the random number from the other party and combines it with its own random number, resulting in session key t :

session key $t = t_u \oplus t_s$
 $= t_u \oplus \text{PVT}_u(\text{PUB}_u(t_s))$
 $= t_s \oplus \text{PVT}_s(\text{PUB}_s(t_u))$

So without exchanging t , both parties have access to it.

Instead of exchanging shared secret s'_i , it is obfuscated with the session key t and exchanged:

$s'_i \oplus t$

Furthermore, for each session, the service obfuscates the already obfuscated key $k_i \oplus s'_i$ with the session key t and stores it in memory:

$(k_i \oplus s'_i) \oplus t$

The service uses double-obfuscated key and obfuscated shared secret to calculate key k_i to the data:

$k_i = ((k_i \oplus s'_i) \oplus t) \oplus (s'_i \oplus t)$

Encrypted communication

For all communication, the transport must be encrypted. We follow the principle that within one system, all transport is encrypted at least once. When it crosses the border of the system, the data must be protected at least twice. So to upload an encrypted forensic image, for example, next to the encryption of the image, the channel to upload the data must be secured as well.

The module to module communication within the service takes place via an encrypted remote procedure call (RPC) framework, where each outgoing and incoming request is automatically logged to the logging service. Communication goes beyond RPC however, and amongst others also includes Hadoop sequence files² and temporary files generated by tools like Apache Tika³ and GroupDocs.⁴

Unique identifiers for images

When uploading a forensic image, the name of the image might reveal details of the case, e.g. the name of the suspect, the name of the case, or the location where material was seized. Therefore, all images uploaded to the central service get a unique randomly generated identifier that is unrelated to the case. Grouping on the file system, if necessary, is done using this identifier and not based on any information related to the case. We do this because even with filenames that show no case related information, grouping images together that belong to the same case may reveal which case the images belong to, e.g. by correlating image sizes to devices and publicly available information.

² <http://wiki.apache.org/hadoop/SequenceFile>.

³ <http://tika.apache.org/>.

⁴ <http://groupdocs.com/>.

User management

Authentication and authorization

It seems obvious that authentication and authorization are crucial to a forensic big data platform. The most basic authorization in HANSKEN is having access to the encryption key (section 3.4). This, however, does not include authentication, making it hard to track who did what in the system. Also, having access to an encryption key doesn't necessarily mean that you are authorized to access every bit of data or perform every function. Some data may be privileged communication (e.g. medical information) that should not be accessible to all investigators.

We do not want to implement a new system where users are administrated and where users have to choose a different password from their existing user accounts. Where possible, we connect to an existing identity manager, e.g. the Active Directory of the organization. This makes it possible to use different types of authentication mechanisms, e.g. simple username/password but also multifactor authentication. Many organizations do not want to open their identity management system to an external system. We have chosen to adapt a model where public keys are exchanged once, and these keys are then used to verify signed requests and responses between HANSKEN and the identity manager. The tokens containing the information about a session, user and its permissions can be carried through the system to make authorization decisions, a so-called *federated identity* (Madsen, Dec. 2005). As a result, all operations within HANSKEN are performed on behalf of a user.

For authorization, apart from having access to the key and shared secret, every function of the system is annotated with a permission. The user needs to have this permission in his token, before he can execute this function. This makes it possible to distinguish between different types of users, e.g. users that can start an extraction process, users that can add tags to a trace and users that can add images to a case.

Secondary identities

Once a user is authenticated, the identity of the user is stored in a token and can be used for logging. This identity is not the core identity of the user. Jericho (Forum, May 2011) dictates us to use secondary identities derived from the core identities. These secondary identities contain enough information to be able to authenticate and authorize the user, but do not reveal more information than needed. So correlating and aggregating activities per user and/or session is possible without the need to know the core identity. Of course, to retrieve who did what on the system, you must have access to the core identity of the user.

Logging framework

Transparency of forensic data means that at any time, it must be clear where traces extracted from the seized material originate from. This keeps the chain of custody in place. All requests to the central service, including authentication and authorization requests, data uploads,

forensic queries and content retrievals are logged to a physically separated logging environment, a separate security domain with its own administrators. Communication with the logging environment is one way: Messages are sent from the service to the logging environment, but it is not possible to send information back from the logging environment to the service.

Logging results in a full trail of who did what with which data, making it transparent, e.g. to the investigation team, to the public prosecutor, to the court or to the suspect's lawyers. Due to the crucial nature of the logging service, the central service must not accept requests if logging is not possible.

Replacement of confidential information

The HANSKEN system will process a lot of private information. This information can be traces extracted from forensic data, e.g. contact information or browser history, but also information about users logging in into the system, e.g. user names and other session details. A lot of this information is stored in logs. There are many different types of users that want to use the logs for different reasons. Administrators use them to monitor the system, auditors use them for auditing, developers can check if any errors occurred during the extraction process and researchers can determine what to focus research efforts on. Even though the log provides this wealth of information, our design principles put a lot of constraints on the information that can easily be read in the log.

We designed a system where all log messages are pre-processed, removing any privacy-sensitive information. It uses a replacement model that, depending on the scope and identifying capabilities, replaces identifying (tagged) information with anonymized (irreversible) or de-identified (reversible) values. Re-identifying (reversing) values can only be done by those who are allowed to, i.e. those who have access to the cryptographic keys needed to reverse the values. Just like the data, the reversible values must be stored encrypted and transport of the log messages must be secured. With access to the valid encryption keys, it is still possible to find out who actually accessed what data and provide accountability. With this model, the more private the information is, the harder it is to reverse this information.

The replacement model, shown in Table 2, consists of 5 different scopes and 2 types of replacement: reversible and irreversible. This combines into 10 different possible replacements. The different scopes are:

Public No specific measures need to be taken to protect this information, since the information is not considered

sensitive. For reversible scope this means that the information is stored as is, for irreversible scope a one way digest is used.

Environment Within a service implementation of the HANSKEN system, the same information should transform into the same result. Within a different service implementation, if the same information is encountered, it should transform into a different result from the first implementation. Examples of this type of information are function calls and host names of services. For reversible, crypto is used with a system wide key. For irreversible, a one way digest is used with a system identifier as salt.

Session Within a user's session, the same information should transform into the same result. If during a new session the same information is encountered, it should transform into a different result. This makes it possible to correlate information within a user session, which can for example help in determining a general work flow, but not extract who the user was or what he or she did. For reversible, crypto is used with a system wide key and the session key as initialization vector (IV). For irreversible, a digest is used with the session key as salt.

Message In the message scope, it is computationally infeasible to correlate information outside of the single log line. The same information within the message can still be correlated: when a value occurs twice within the message, it is transformed into the same result. For reversible, a unique id is generated. This id is stored in the log and the original value is stored in an external look-up table. Same values within the message are assigned the same id. For irreversible, a simple unique number per value is assigned.

None Every single value is considered sensitive and it should never be possible to correlate the information if you don't have access to the original material. For reversible a unique identifier is assigned per value, even if two values are the same. For irreversible, the value can simply be removed from the log.

The design principles dictate that *None Irreversible* is the default replacement, so when no scope is defined, the value belonging to this tag is removed from the log. Assigning a transformation to a tag can be done separately from the implementation and can be changed according to new insights, legislation or business needs.

As an example the following logline:

```
[user>harm] performed query [query>keyword_query] [keyword>elephant] on project [project>diamond]
```

Table 3 shows the transformations that are assigned to the different tags.

Applying these transformations to the logline results in the following:

Table 2
Replacement model for privacy sensitive information.

Scope	Reversible (de-identified)	Irreversible (anonymized)
Public	P R	P I
Environment	E R	E I
Session	S R	S I
Message	M R	M I
None	N R	N I

Table 3
Example transformations.

Tag	Transformation
User	Environment Reversible (ER)
Query	Public Reversible (PR)
Keyword	Session Irreversible (SI)
Project	Message Reversible (MR)

```
[user>ZWQyNjM3Njg=] performed query [query>keyword_query] [keyword>033a8c53] on project [project>1]
```

Tool responsibility

During the extraction process, tools are applied from forensic libraries (see section 3.1). To be able to determine the origin of a trace, it must be clear which tools accessed the data, what traces resulted from applying the tools to the data and what properties were created by which tools. If a tool fails during the extraction of a trace, this should also be stored and queryable.

Business rules

In a multi tenant environment, resources are shared. To manage resources and prioritize jobs, we set up a dedicated orchestration service. This service applies rules set by the business to prioritize jobs. For example, a representative of the public prosecutor can define that the extraction of traces from images related to child abuse cases take precedence over the extraction of traces from images related to burglaries. Other examples of factors that may influence job prioritization are the originating organization, current load on the system, types of tools to be applied and manual priority.

Best coding practices

The eight principles are also upheld by embracing development standards, e.g. Scrum-framework,⁵ code reviews and extensive testing via unit tests, integration tests and regression tests.

Open standards

We use open standards where possible. These standards include cryptographic algorithms, message transport protocols, file storage formats, job distribution, cluster management, etc. By using open standards, we make sure we are not locked in to a specific vendor and have the ability to replace parts of the implementation. Furthermore, using open standard makes it possible to use software that implements these standards and is maintained by vendors or communities.

Separation of concerns

Apart from using open standards, we apply the design principle *separation of concerns*. This means that the implementation is split into multiple modules that all implement correlated functionality. Each module provides an interface that can be used by other modules (comparable to the façade design pattern), making it possible to better use, integrate and test modules and easily replace module implementations.

Single point for external access

External access to the service is provided via a separate module as well. This module provides a RESTful API

(Fielding, 2000), such that clients can communicate with it. This interface serves as a base for separately developed graphical user interfaces and scripts. This module is based on XIRAF's query language that is currently in use.

No single point of failure

The service should not contain a single point of failure (SPOF). This means that the system should not depend on one single machine: if a single machine fails, the system must continue servicing its full functionality. Therefore, we use distributed technologies. Many implementations exist for the requirements we want to implement: distributed storage, distributed processing of data and a distributed search engine.

Implementation

Keeping in mind the motivation and design considerations from the previous two sections, we have implemented a system. The data in this system is protected, privacy measures with respect to the data and its users are taken into account and it is transparent who did what with the data. Designing a service also involves hardware, network configuration and software. In this section we discuss the impact of the principles to the software solution and system design itself.

At a high level, the system consists of multiple cooperating *independent modules* that all implement correlated functionality. This section describes the different modules, the implementation of these modules and the communication between the modules. Where applicable, the considerations described in Section 3 are mentioned. Since not every consideration relates to a single module but can span multiple modules, they are mentioned where most applicable.

Although a lot of functionality is already implemented in HANSKEN, not every module is completed. Missing functionality is described as we foresee it.

Modules

Fig. 2 shows an overview of the modules the HANSKEN system consists of, including their intermediate communication and the basic technologies they are implemented on. In the HANSKEN system, we call the modules *services*: they all serve specific functionality.

RPC framework

One of the most fundamental components of HANSKEN is the *RPC Framework*. This framework is the basis of communication between the different modules and contains a lot of the shared functionality of these modules.

Fig. 3 shows the RPC stack. The first responsibility of the framework is setting up communication, including a failover mechanism (consideration 12). This mechanism is implemented in two different ways: static failover, where failover hosts are preconfigured, and dynamic failover, where failover hosts are registered in Zookeeper.⁶

⁵ <https://www.scrum.org/>.

⁶ <http://zookeeper.apache.org/>.

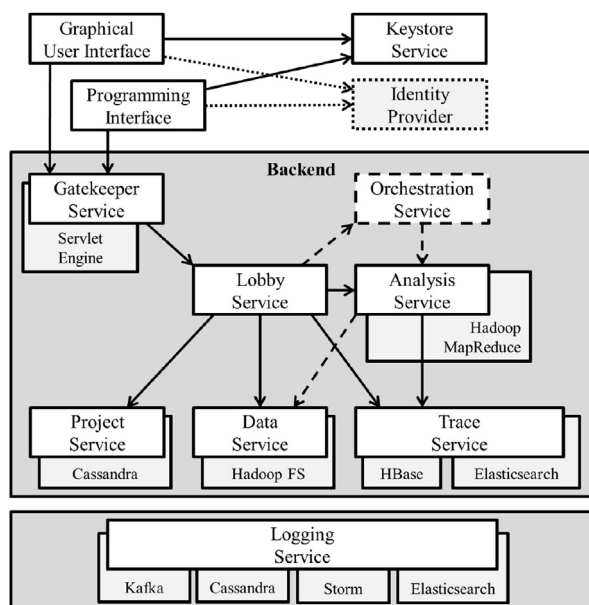


Fig. 2. Module overview.

The next responsibility of the framework is logging everything that is communicated (consideration 8). If a module sets up communication with another module and sends a message, this results in at least six log messages: two log messages for setting up communication (one at each module), two log messages for the sending the request message and two log messages for the response message.

Another responsibility of the RPC framework is distributing information that is required in pretty much every module. This includes distributing credentials. For authentication and authorization (consideration 7) we use the SAML 2.0 standard.⁷

Since Jericho dictates that every module should protect itself, authorization must take place at the module. The authorization mechanism is implemented in the RPC framework as well, which eliminates the need for a separate authorization service. Each public function on a service is annotated with required permissions, possibly with a identifier for a specific object, e.g. a project (case) or an image. The provided credentials must contain the right permissions in order to execute the function.

The current implementation of the RPC Framework uses the Netty Framework⁸ to send the bytes across the wire.

Gatekeeper Service

The *Gatekeeper service* is the module that communicates with the outside world. The first responsibility of this module is related to authentication (consideration 7). As mentioned in Section 4.1.1, we use the SAML 2.0 standard for authentication and authorization. The Gatekeeper acts as a Service Provider (SP). When a user is not yet authenticated when accessing HANSKEN, the Gatekeeper service redirects the user

to an identity provider (IdP), which is responsible for authenticating the user. The identity provider is not part of HANSKEN. Any identity provider that is able to provide SAML-tokens suffices, e.g. Active Directory. By outsourcing authentication, organization can choose their own type of authentication mechanisms. Furthermore, it opens up the possibility for single sign-on. The Gatekeeper puts the user credentials in the RPC-request to be used throughout HANSKEN.

The Gatekeeper provides a RESTful web service (Fielding, 2000) (consideration 12). All functionality implemented in HANSKEN must be available through this interface, like searching, creating projects and starting the extraction process. The module translates these requests to RPC-requests and communicates them to the Lobby Service.

Lobby Service

The *Lobby Service* redirects user calls to the appropriate modules. It is aware of the different routes that function calls should follow and makes these calls in appropriate order. Search queries for example are typically performed on projects (a collection of images). It is the responsibility of the Lobby Service to first retrieve the list of images from the Project Service and send the query on these images to the Trace Service.

Orchestration service

The *Orchestration Service* is responsible for making business decisions based on a set of rules (consideration 11). These business rules are defined and maintained outside the module and determines priority for different functions based on these rules. Although this module is not implemented yet, we have done some experiments with Drools⁹ and the results are promising.

Project Service

The *Project Service* is responsible for storing information related to images and cases (which we call projects). Images are stored with a de-identified name on the file system (consideration 6). The unique identifiers need to be transformed into names that make sense to human investigators. A case normally consists of multiple images. This module administers which images are combined into cases. So, the Project Service administers images and projects, including details about these objects, like the name of the person and location where a device was confiscated, the name of a case and the name of the investigator that created an image.

The Project Service is implemented on top of a simple key-value store. Current implementations offer both serialization to disk using Kryo¹⁰ and storing the image and project details in a Cassandra store.¹¹

Data service

The *Data Service* is responsible for retrieving data from images (consideration 3). The current implementation uses

⁷ <https://www.oasis-open.org/standards#samlv2.0>.

⁸ <http://netty.io/>.

⁹ <http://www.drools.org/>.

¹⁰ <https://github.com/EsotericSoftware/kryo>.

¹¹ <http://cassandra.apache.org/>.

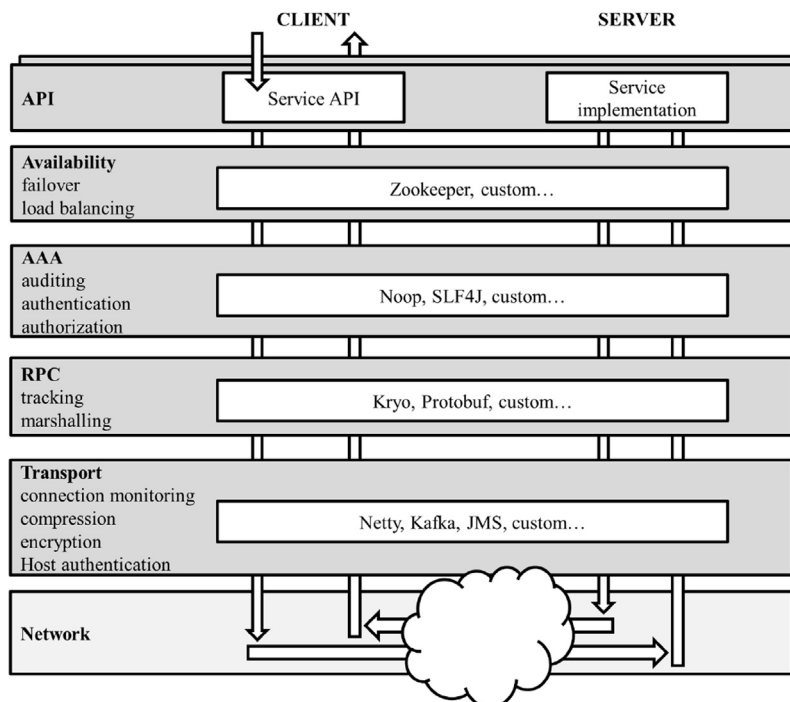


Fig. 3. RPC framework.

a hybrid model. On the one hand it is possible to run the Data Service as a standalone service like any other module in HANSKEN, on the other hand it is possible to embed the Data Service in another module. This is done for performance reasons. For relatively infrequent calls, like showing a picture in a GUI, it is feasible to read data using an RPC-call. This includes all the benefits as described in Section 4.1.1. For the extraction of traces from an image, however, the number of reads and the amount of data to be read is currently too large to retrieve via our current RPC implementation. The downside of including the Data Service as a module is that we don't get the benefits of the RPC-framework like logging of all the calls. It also contradicts the Jericho principles that every service should protect itself: the Data Service has to trust the Extraction Service (see Section 4.1.8) with the key of the data in order to provide the data. When external parties add tools to the Extraction Service, these tools have to be trusted with the key to the data as well.

For now, to trust these parties with the key to the data, we prefer making the tools an integral part of HANSKEN, including a code review to check for potential data leaks (consideration 12).

In the future we want to be able to only run the Data Service as a service, where the Extraction Service can communicate using channels that have less overhead than RPC on TCP/IP but still allows for the same benefits, e.g. RPC on Unix sockets.

To read data from an image, the Data Service requires a number of parameters. Other than the offset, the size and the key, the service requires a transformation path to be able to read the data as it was originally read. This means that to retrieve the contents of an attachment in a PST-

mailbox, the Data Service needs to know the location of the PST-file on disk, the type of PST-encryption used (none, Permutation or Cyclic (Microsoft Corporation, 2014)) and where the attachment resides in the PST-file. These transformations are generated during the extraction and stored with the traces in the Trace Service in a serialized format (currently using JSON Smile,¹² but other serializations are possible). When retrieving the data of a trace, this transformation is provided to the Data Service and used to retrieve the original data. This makes it possible to prevent data being copied out to temporary files when retrieving or processing it (consideration 5).

For performance, scalability and high availability we use a distributed file system for storing the data. We have chosen the Hadoop Distributed File System (HDFS)¹³ (Shvachko et al., 2010), due to its natural connection to MapReduce (Dean and Ghemawat, 2008). Apart from that, we implemented a version that runs on top of a local file system.

Keystore Service

The *Keystore Service* is responsible for storing the encrypted obfuscated keys and encrypted shared secrets (consideration 4):

$PUB_u(s'_i)$: user part of the key

$PUB_S(k_i \oplus s'_i)$: service part of the key

¹² <http://wiki.fasterxml.com/SmileFormat>.

¹³ <http://wiki.apache.org/hadoop/HDFS>.

The module that provides the public key for the system is the Data Service (Section 4.1.6). As shown in Fig. 2, the Keystore Service is preferably not deployed within the HANSKEN service, but in a separate security domain. This means that with each call to retrieve or process data, the user has to provide both keys before the Data Service can actually read data from the image. Consequently, users only have access to images that they have been explicitly granted access to.

Current implementation of the Keystore Service provides an RPC-interface to add, retrieve and delete keys and has options to store the keys in memory or a Lucene index.¹⁴

Extraction Service

The responsibility of the *Extraction Service* is to analyze the data and extract traces from it. It applies tools from forensic libraries (consideration 1) to the data and sends the resulting trace information to the Trace Service (section 4.1.9). The tools range from parsing file systems and files to carving unallocated space and extracting keywords.

In the proof of concept we implemented in 2012, we based the forensic extraction process on the Hadoop implementation¹⁵ of MapReduce (Dean and Ghemawat, 2008). MapReduce is a framework for distributed processing of data (consideration 2). Although using MapReduce drastically speeds up the extraction process (from 24 h per terabyte in XIRAF to over 3 terabytes per hour in our proof of concept implementation), MapReduce is a batch-based process. To extract all traces from a forensic image, we still need about three to five iterations. This is caused by the fact that in our current implementation the file system traversal cannot be parallelized. The MapReduce process for parallel processing of all files in a file system can only start once the file system traversal is finished. We are currently doing research on processing extracted traces as soon as they become available from any tool. For this, we plan to use a streaming extraction like Storm¹⁶ on top of a Kafka queue.¹⁷

In the MapReduce implementation (and the planned Storm implementation), the traces are available for querying soon after they are extracted from the forensic image. No additional publication is needed for this. Although not all data is available, this gives investigation teams the opportunity to have access to traces early in the investigation. This is especially true when this functionality is combined with the dynamic pipelining of the extraction process and the imaging process, as explained in Section 3.2.

The order in which tools are currently applied is iterative, based on a statically ordered list. We use the rationale behind smoothsort (Dijkstra, 1982) to make sure that the tools are applied in the correct order in only a few iterations. This means for example that in the static list of tools one of the first tools to be applied is the tool to determine

the mimetype. This is because a lot of sequential tools use this information to determine if they can run on this object. Another example is that the tool to calculate hashes has to run before the tool to check if a hash is present in a hash database. Since no image is the same and the optimal order of tools is different for each image, we are currently satisfied with this solution. In the future we want to automatically determine an even more optimal ordering for the tools instead of the current best effort, but the expected time it takes to build this mechanism versus the gain in extraction speed does not (yet) justify the resource use.

We do not yet support the ability to run additional, custom, tools. It is possible for external parties to create tools and add them to the Extraction Service. These tools however need to be part of the Extraction Service during the extraction process in order to be applied. Currently, it is not possible for an investigator to run a custom MapReduce job where he uses picture analysis to search all pictures for a specific object, for example. In the future, a user should be able to run a customized job and store the results in the Trace Service. This functionality is referred to as asynchronous queries.

Trace Service

The *Trace Service* is responsible for storing and retrieving traces. Traces consist of metadata, a full keyword index and (a link to) the actual data of the trace. For real time querying, a distributed search engine fits better than a distributed data store like a key-value store or document store (Ugen, April 2013). For this, we use Elasticsearch.¹⁸ Preparing for asynchronous queries, we also store the traces in HBase.¹⁹

Every trace stored in the Trace Service has one or more types. The properties of the trace are determined by a set of base properties, e.g. a unique identifier, type and name, combined with additional properties for the types of a trace, e.g. modification date, e-mail subject or phone number. Additionally, for each trace we store information about the tools that were applied to the trace (consideration 10). Therefore, with all properties of all traces we store the tool that extracted it, both its unique name and version, and if the tool was applied successfully. We also store the time the trace was published by the system. This information can be used to search the logs for any additional information about the trace.

Traces can have data associated with it. As described in Section 4.1.6, a transformation path is stored with these traces. A trace can have multiple data streams: a mail message stored in a PST-mailbox can be stored in different formats (Microsoft Corporation, 2014): preview, plain text, RTF compressed and HTML. Messages generally have more than one stream and need to be stored accordingly. The same is true for Word documents: we store both the raw stream, the bytes as they are stored on disk, but also the text stream, without the additional metadata. By storing these different data streams, we determine the different data properties (hashes, entropy, mimetype) and allow for

¹⁴ <http://lucene.apache.org/>.

¹⁵ <http://wiki.apache.org/hadoop/MapReduce>.

¹⁶ <http://storm-project.net/>.

¹⁷ <http://kafka.apache.org/>.

¹⁸ <http://www.elasticsearch.org/>.

¹⁹ <http://hbase.apache.org/>.

additional automatic analysis, e.g. linking the text stream of a Word document to the plain text stream of a PST e-mail message. Fig. 4 shows an example of a trace.

Traces can be queried in a number of ways. All the query options described in Bhoedjang et al. (2012) are available,

including searching for keywords, querying property values and creating (super) time lines. Moreover, ElasticSearch implements aggregations, meaning that together with a query result, statistics on the results are provided. This functionality gives a boost to the graphical user

Id:	dd983a06-da72-4d25-ab7e-abb8582c783c:0-0-0-19-6-0-3-0-3
Path:	/dd983a06-da72-4d25-ab7e-abb8582c783c/0///mail/pst/outlook-2003-unicode-unencrypted.pst/Inbox/ Subfolder of Inbox/RAR library
data	
plain	
entropy	3.277225302470384
hash	
md5	72b51a7c428f74c91b71d7c3d5e8e273
sha1	9f9391e1db7a2e3d91122340ab7a06c3f07b8b9d
sha256	c365108a8edce51805a5d3a8f8351b45a3d9ea5be82b1716efbd1d71217dicf3
mimeType	application/octet-stream
size	856
preview	
entropy	3.249847495704546
hash	
md5	0d5e6b984ab36fe02218186d573d5b2b
sha1	8045033b19f835a1e4a44f74515f428d2c37c2ac
sha256	e52a1c89d70cd8a62278dd0bbe4a82ccf8109f0a8efd0976d75f1a8bc85f9e88
mimeType	application/octet-stream
size	510
rtf	
entropy	6.713648848534523
hash	
md5	885589ced362fb314f5c1497267ab939
sha1	6e71aaed53e036bca8a061277ca76cbea04ec43e
sha256	e31bd9d6fcb8087ea321532de872bee136daf782ac07cfbff51b59b03cd03d2
mimeType	application/octet-stream
size	494
email	
application	OUTLOOK
from	...
hasAttachment	true
messageId	<F3B44E70E39E094DA5E637D07D02A83C014D1DC6@...>
read	true
receivedOn	2010-05-18T15:42:45.163Z
sentOn	2010-05-18T15:42:45.163Z
subject	RAR library
to	...
tool	
meta	
creator	traces/email
properties	
data/digest/entropy	data.plain.entropy, data.rtf.entropy, data.preview.entropy
data/digest/md5	data.plain.hash.md5, data.rtf.hash.md5, data.preview.hash.md5
data/digest/sha1	data.plain.hash.sha1, data.rtf.hash.sha1, data.preview.hash.sha1
data/digest/sha256	data.plain.hash.sha256, data.rtf.hash.sha256, data.preview.hash.sha256
traces/email	email.application, email.hasAttachment, email.read, email.subject, email.messageId, email.sentOn, email.receivedOn, email.modifiedOn, email.to, email.cc, email.bcc, dataTypes, data.plain.descriptor, data.plain.size, dataTypes, data.rtf.descriptor, data.rtf.size, dataTypes, data.preview.descriptor, data.preview.size
traces/mime	data.plain.mimeType, data.rtf.mimeType, data.preview.mimeType
publishedOn	2015-02-12T13:24:08.738Z
success	traces/mime
types	
data	traces/email
email	traces/email
version	
data/digest/entropy	0.1
data/digest/md5	0.1
data/digest/sha1	0.1
data/digest/sha256	0.1
data/text	1.0
traces/email	1.0
traces/mime	1.0
plain	
creator	traces/email
success	data/digest/entropy, data/digest/md5, data/digest/sha1, data/digest/sha256, data/text
preview	
creator	traces/email
success	data/digest/entropy, data/digest/md5, data/digest/sha1, data/digest/sha256, data/text
rtf	
creator	traces/email
success	data/digest/entropy, data/digest/md5, data/digest/sha1, data/digest/sha256, data/text

Fig. 4. Trace example.

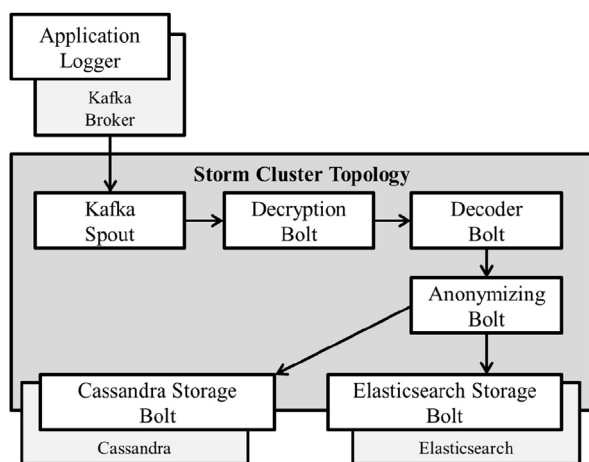


Fig. 5. Logging service.

interface, giving much better insight in the traces extracted and thus in identifying significant seized material.

Logging service

When HANSKEN is running, a lot of log messages are generated (consideration 8). Examples are log messages generated by user activity, the extraction process and communication, but also includes log messages generated by the operating system and the forensic libraries. Rough estimates suggest that the number of log messages generated in The Netherlands will be in the range of 100,000 per second. This is a number not uncommon in a lot of big data systems and software is available to handle these numbers. We have chosen to adapt a Kafka/Storm-cluster for our *Logging Service*. Fig. 5 shows the different parts of this service.

Logging starts by capturing the log messages using a logger. We use SLF4J²⁰ for the API. Most applications log their messages using Log4J.²¹ The appender is responsible for encoding and encrypting the log message. Our current implementation uses a fixed hard coded encryption key for the logging framework. Since this key can be compromised, we want to migrate to a more secure solution, like FI-BAF (Yavuz et al., 2012a) or LOG-FAS (Yavuz et al., 2012b).

After the message is encoded and encrypted, it is put on a Kafka-queue. Kafka is very robust and can handle a large number of messages. They are persisted to disk and can be archived if needed. Since the messages are encrypted, this does not impact security. A Kafka spout²² retrieves the messages from the queue and sends them to the first bolt in the Storm cluster. This bolt decrypts the message and sends in to the decoder bolt. The next bolt anonymizes the message where applicable (consideration 9). The information that needs to be stored in the external look-up table is sent to the CassandraStorageBolt, which stores it in a Cassandra

store. The anonymized log messages are sent to a bolt that stores them in an Elasticsearch cluster.

Additional bolts can be added where needed, e.g. to validate log messages, to send alerts to administrators or developers if a certain log message passes through or to combine log messages to a higher level message.

Service implementation

The modules have to run on hardware and software that impact the chosen solutions as well. Apart from trusting users, (system) software needs to be trusted as well. A server (host) that is added to the environment has to authenticate itself. It must be enforced that the server is trusted and allowed to store forensic images, store logging information or take part in the distributed data search engine.

For implementing trust relations between hosts and services we use basic Kerberos (Neuman and Ts'o, 1994). To assure that a host is allowed to join a service, the host needs to provide a ticket. This ticket is provided by the Authentication Server which is part of the Key Distribution Center.

Apart from the examples above, the principles have more impact on deployment. Examples are server virtualization, encryption of temporary files like MapReduce sequence files and encryption of stored databases. All this has been addressed in our current configuration. However, discussing the deployment of HANSKEN in detail goes beyond the scope of this paper.

User interface

Since HANSKEN is designed as a service with an open interface, an organization can design and implement its own user interface according to its own needs. We have implemented a number of interfaces according to our needs and based on we have learned from our users over the number of years we run XIRAF. The user interfaces are not the subject of this paper, but for completeness we briefly show the implemented functionality.

Graphical user interfaces

We have developed two graphical user interfaces: A tactical user interface, aimed at detectives (van Baar et al., 2014), and a technical interface, aimed at digital investigators. Both interfaces provide the same functionality, however the tactical interface is more explicit in the possible queries that can be performed and shows less technical details, e.g. it misses a hex view.

The tactical interface is implemented on top of XIRAF and is compatible with HANSKEN. The technical interface is newly designed, since the interface shown in Bhoedjang et al. (2012) is not compatible with HANSKEN. Fig. 6 shows the basic search page where the user searched for e-mails containing the term peter.

The technical user interface is currently being designed and implemented. The main difference with the tactical user interface is that the technical interface gives direct access to all features of the query language, presentation of the results is configurable (based on a sortable table where the user can select and order columns) and more in-depth

²⁰ <http://www.slf4j.org/>.

²¹ <http://logging.apache.org/log4j/>.

²² <https://github.com/HolmesNL/kafka-spout>.

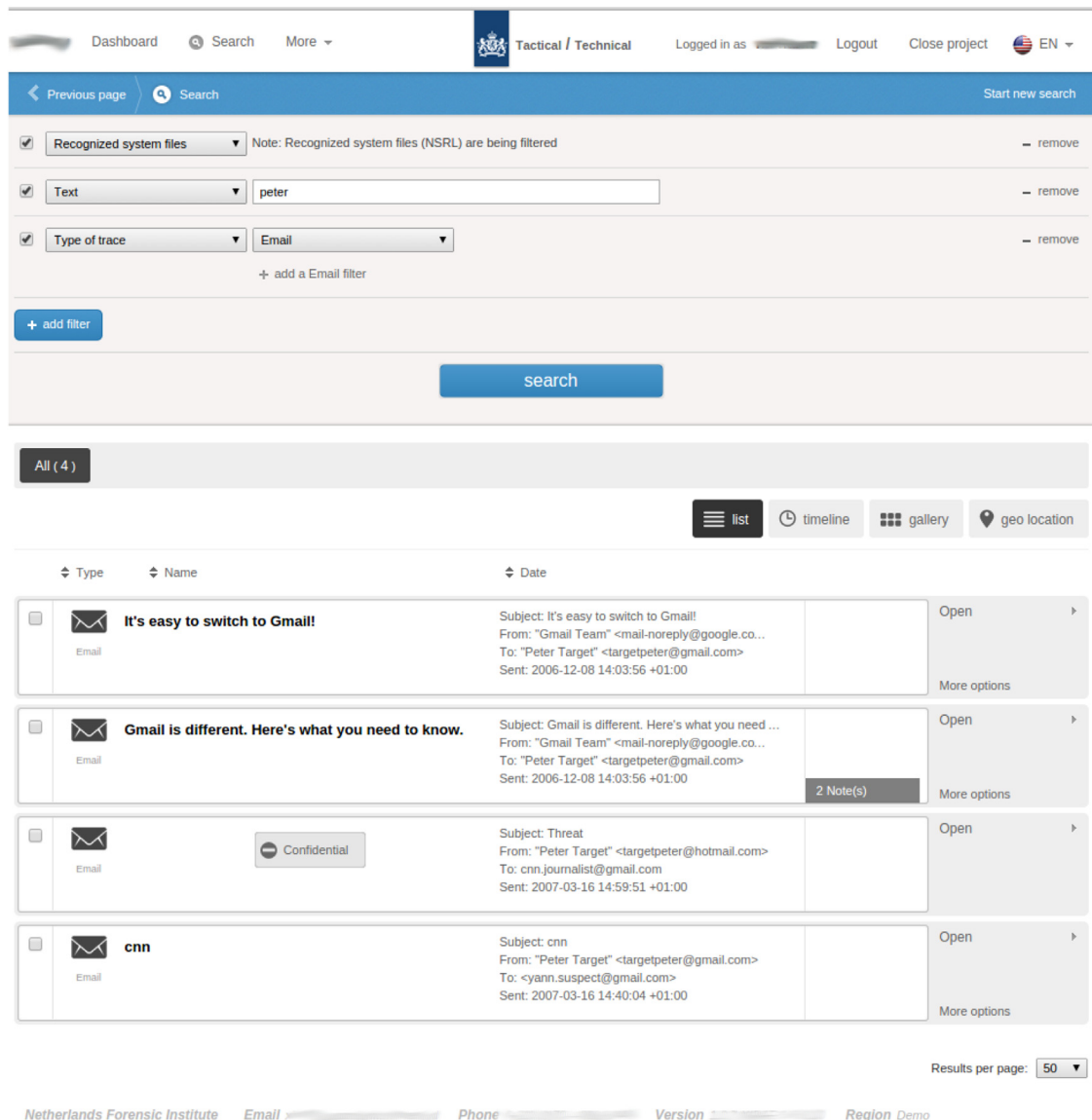


Fig. 6. Tactical user interface.

views of the data are available, like a hex viewer. Fig. 7 shows a preview of the technical user interface.

Programmatic user interfaces

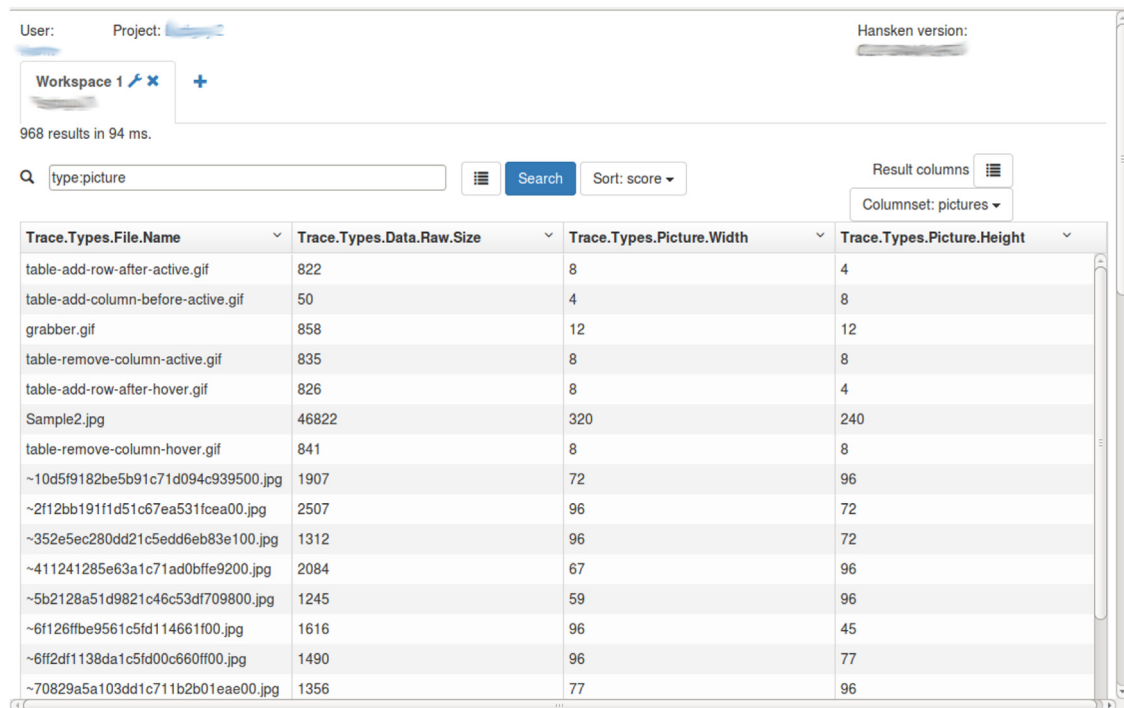
Next to graphical user interfaces we also provide bindings that can be used to connect to HANSKEN using a programming language. We have currently implemented bindings in Python, making it possible to write scripts in Python or connect using an interactive shell and query the database.

Related work

Quick and Choo (Quick and Choo, 2014) present an overview of the impacts of increasing volume of digital forensic data. They summarize the published research and

future research challenges with respect to this topic. They state that there is a need for real world applicability of methods to address the digital forensic data volume challenge.

A way to cope with the new big data challenges is to perform triage, a term borrowed from emergency response in which patients are prioritized according to their likelihood of survival. With triage it is possible to make an educated guess on whether or not seized material contains relevant traces. This makes it possible to quickly reduce the big amount of material and decrease the processing time. A lot of work is done in the field of triage, e.g. by Roussev et al. (Roussev et al., 2013) and Garfinkel (Garfinkel, 2013). Triage is a valuable approach which we plan to use for ordering the processing of images, *not* for leaving images unprocessed.



Trace.Types.File.Name	Trace.Types.Data.Raw.Size	Trace.Types.Picture.Width	Trace.Types.Picture.Height
table-add-row-after-active.gif	822	8	4
table-add-column-before-active.gif	50	4	8
grabber.gif	858	12	12
table-remove-column-active.gif	835	8	8
table-add-row-after-hover.gif	826	8	4
Sample2.jpg	46822	320	240
table-remove-column-hover.gif	841	8	8
~10d5f9182be5b91c71d094c939500.jpg	1907	72	96
~2f12bb191f1d51c67ea531fcea00.jpg	2507	96	72
~352e5ec280dd21c5edd6eb83e100.jpg	1312	96	72
~411241285e63a1c71ad0bffe9200.jpg	2084	67	96
~5b2128a51d9821c46c53df709800.jpg	1245	59	96
~6f126ffbe9561c5fd114661f00.jpg	1616	96	45
~6ff2df1138da1c5fd00c660ff00.jpg	1490	96	77
~70829a5a103dd1c711b2b01eae00.jpg	1356	77	96

Fig. 7. Preview of the technical user interface.

Multiple next generation forensic analysis systems are under development or in production. These systems are generally built to automate and speed-up the extraction of traces from forensic images, which is a good starting point to set up Digital Forensics as a Service (DFaaS). In 2004, Roussev et al. (Roussev and Richard, 2004) described a distributed processing system many times faster than FTK.²³ This was a lab setup. Since then FTK 3 and higher support a total of four so-called workers to automatically process data in parallel. Research on the automated processing of seized material was coined in 2006 by Alink et al. (2006). Ayers (2009) put down the need for such a system and described the requirements that such a system must, should or may meet. In 2012, Bhoedjang et al. (2012) explained how the XIRAF system is engineered and in use in the Netherlands. One of the efforts to build a DFaaS system is proposed by Lee and Un (Lee and Un, 2012). They focused on speed and provided the end-user with a web interface to search through the data. Cohen et al. (Cohen et al., 2011) present the Google Rapid Response (GRR) framework, a platform for enterprise forensic investigations enabling remote raw disk and memory access. Via this framework, multiple remote machines can be acquired and analyzed concurrently. Where our focus lies in extraction traces from seized (off line) devices, they focus on (on line) corporate settings. Wen et al. (Wen et al., 2013) introduce a service-based framework for supporting computer forensics work flow, based on GRR. They designed a

cloud-based framework for dealing with large volume of forensic data, sharing inter-operable forensic software, and providing tools for investigators to create and customize data processing work flow. They propose a “forensic app store” where work flow can be constructed using software components (“apps”). Forensic examiners and investigators can on-demand create, invoke, and deploy tasks based on the functionality available in the software components. Recently, Cruz et al. (Cruz et al., 2015) propose a new distributed data store for remote forensics that partitions data into database files that can be accessed independently so that distributed forensic analysis can be done.

In 2002, Carrier (Carrier, October 2002) set out that legal arguments must be taken into account when developing forensic tools. He concluded that open source tools may more clearly and comprehensively meet the guidelines than closed source tools do. In our opinion, the code does not need to be open as long as the tests are clear and others than the developers have the possibility to run their tests against (parts of) the code. From that perspective, transparency is more important than providing the code itself.

A lot of work is done in the field of encrypting data. Song et al. (Song et al., 2000), for example, present a technique for searches on encrypted data. Their solutions support searching data that is not known to the organization that stores the data. The solution they provide focuses on exact text searches, which is too narrow for the solution we propose.

Bayuk (Bayuk, 2011) recommends a systems-level approach to security validation of a cloud service.

²³ <http://www.accessdata.com/products/digital-forensics/ftk>.

Jansen and Grance of NIST (Jansen and Grance, December 2011) have written a comprehensive set of guidelines for the outsourcing of data, applications and infrastructure to a public cloud environment, while acknowledging that the entire cloud ecosystem is in rapid development. Their primary recommendation is: “Carefully plan the security and privacy aspects of cloud computing solutions before engaging them”, which corresponds to the call by the Canadian Information and Privacy Commissioner (Ann Cavoukian, January 2013) to integrate good privacy and security measures into information systems early on.

Aminnezhad et al. (Aminnezhad et al., 2012) give an extensive overview of how privacy is and can be handled in digital forensics. They conclude that privacy measures should be part of the design and not added as counter-measures in a later stage. Apart from that, they note that there should be more focus on education and awareness of preserving privacy in the professional forensic field.

Law et al. (Law et al., 2011) state that to protect privacy, personal data that are not related to the investigation subject should be excluded during computer forensic examination. They propose a procedure for handling private digital data, based on an encryption key that is kept by the owner.

Shebaro (2012) describes how forensic tools can be developed that preserve the privacy of the persons in network data, while at the same time enabling law enforcement to perform an investigation.

Tan et al. (Tan et al., 2013) review solutions that address system security and data provenance in distributed systems. They argue that forensics of a cloud solution is only possible if that data is trustworthy, for which data provenance is a necessity.

Conclusions

In this paper, we motivate why a centralized service for large scale forensic data analysis is a good idea from a business perspective. The forensic drivers for this are (1) minimization of case lead time, meaning that the data should be available to investigation teams as soon as possible, (2) maximization of the coverage, meaning that we want to understand as much from the seized digital material as possible, and (3) specialize people, meaning that dedicated people have specialized tasks for which they are educated and equipped.

Centralization of forensic data analysis and the associated risks, mandate keeping track of several design principles; we identified eight. The three most important principles are sociologically driven and go hand-in-hand: security, privacy and transparency. We set out these principles from the viewpoint of the seized material, the people involved with processing the data, and system design itself. The other five principles are mainly business driven: multi tenancy, future proof, data retention, reliability and high availability.

From the drivers and principles, we extracted a total of twelve topics that are most important for the design of the HANSKEN system: reuse of forensic knowledge, distribution of the extraction of traces from forensic images (bring the

tools to the data and let the data drive the analysis and extraction), the image format used to store the data, key management (creation, storage and exchange), encryption of all communication, the use of unique identifiers for the forensic images, user management (authentication and authorization and the use of secondary identities), the logging framework, de-identification or anonymization of confidential information in the logs, responsibility and traceability of the tools that analyze the data and extract traces, business rules for orchestrating the system and finally our coding practices (use of open standards, separation of concerns, access via a REST interface and no single point of failure).

In this paper we briefly addressed these topics and explained how they impact the design and implementation of the HANSKEN system. The system consist of several modules that all serve a specific goal: the Gatekeeper Service as central entrance point, handling authentication, the Lobby Service as central unit for routing all incoming requests, the Orchestration Service for business decisions, the Project Service for storing data related to images and cases, the Data Service for providing access to the data, the Keystore Service for storing keys, the Extraction Service for applying forensic tools to the data to extract traces, the Trace Service for storing and querying the extracted traces and finally the Logging Service for handling and storing all logs generated by the other services. We explained how all these modules intercommunicate using RPC and what (distributed) technologies they are based on.

Although the HANSKEN system is still under development, the first cases are run with it. We plan to replace XIRAF by HANSKEN before the end of this year.

References

- Alink W, Bhoedjang R, Boncz PA, de Vries AP. Xiraf—xml-based indexing and querying for digital forensics. *Digit Investig* 2006;3:50–8.
- Aminnezhad A, Dehghantanha A, Abdullah MT. A survey on privacy issues in digital forensics. *Int J Cyber-Secur Dig Forensics (IJCSDF)* 2012;1(4): 311–23.
- Ann Cavoukian MC. Privacy and security by design: a convergence of paradigms. Tech. rep. Office of the Information and Privacy Commissioner; January 2013.
- Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, et al. A view of cloud computing. *Commun ACM* 2010;53(4):50–8. <http://dx.doi.org/10.1145/1721654.1721672>.
- Ayers D. A second generation computer forensic analysis system. *Digit Investig* 2009;6:S34–42. <http://dx.doi.org/10.1016/j.diin.2009.06.013>.
- Bayuk J. Cloud security metrics. In: *System of Systems Engineering (SoSE)*, 6th International Conference on, 2011; 2011. p. 341–5. <http://dx.doi.org/10.1109/SYSSOE.2011.5966621>.
- Bhoedjang RAF, van Ballegooij AR, van Beek HMA, van Schie JC, Dillema FW, van Baar RB, et al. Engineering an online computer forensic service. *Digit Investig* 2012;9(2):96–108. <http://dx.doi.org/10.1016/j.diin.2012.10.001>.
- Carrier B. Open source digital forensics tools: the legal argument. Tech. rep., @stake Research Report. October 2002.
- Cohen M, Garfinkel S, Schatz B. Extending the advanced forensic format to accommodate multiple data sources, logical evidence, arbitrary information and forensic workflow. *Digit Investig* 2009;6(Suppl. 0): S57–68. the Proceedings of the Ninth Annual DFRWS Conference, <http://dx.doi.org/10.1016/j.diin.2009.06.010>, <http://www.sciencedirect.com/science/article/pii/S1742287609000401>. URL.
- Cohen M, Bilby D, Caronni G. Distributed forensics and incident response in the enterprise. *Digit Investig* 2011;8:S101–10.
- Cruz F, Moser A, Cohen M. A scalable file based data store for forensic analysis. *Digit Investig* 2015;12(Suppl. 1):S90–101. DFRWS 2015 Europe Proceedings of the Second Annual DFRWS Europe. doi, <http://>

- dx.doi.org/10.1016/j.diin.2015.01.016, <http://www.sciencedirect.com/science/article/pii/S1742287615000171>. URL.
- Dean J, Ghemawat S. Mapreduce: simplified data processing on large clusters. *Commun ACM* 2008;51(1):107–13. <http://dx.doi.org/10.1145/1327452.1327492>.
- Diffie W, Hellman M. New directions in cryptography. *IEEE Trans Inf Theory* 2006;22(6):644–54. <http://dx.doi.org/10.1109/TIT.1976.1055638>. URL, <http://dx.doi.org/10.1109/TIT.1976.1055638>.
- Dijkstra EW. Smoothsort, an alternative for sorting in situ. In: *Theoretical foundations of programming Methodology*. Springer; 1982. p. 3–17.
- Fielding RT. Architectural styles and the design of network-based software architectures. Ph.D. thesis. Irvine (: University of California; 2000.
- Forum Jericho. Jericho forum commandments, version 1.1. Dec. 2006.
- Forum Jericho. Jericho forum identity commandments, version 1.0. May 2011.
- Garfinkel SL. Digital media triage with bulk data analysis and bulk_extractor. *Comput Secur* 2013;32:56–72.
- Humble J, Farley D. Continuous delivery: reliable software releases through build, test, and deployment automation. Addison-Wesley Signature Series (Fowler), Pearson Education; 2010. URL, <http://books.google.nl/books?id=6ADDuzere-YC>.
- ISO. Information technology – security techniques – information security management systems. Geneva, Switzerland: ISO 27001, International Organization for Standardization; 2013.
- Jansen W, Grance T. Guidelines on security and privacy in public cloud computing. Tech. rep. Gaithersburg, MD: National Institute of Standards and Technology (NIST); December 2011.
- Joyce T. Closing the case: solving violent crimes quickly and efficiently with public records. *Police Chief* 2012;79:50–6. URL, http://www.policechiefmagazine.org/magazine/index.cfm?fuseaction=display_arch&article_id=2604&issue_id=22012.
- Kohn M, Eloff M, Eloff J. Integrated digital forensic process model. *Comput Secur* 2013;38(0):103–15. <http://dx.doi.org/10.1016/j.cose.2013.05.001>.
- Lacey D. Inventing the future – the vision of the jericho forum. *Inf Secur Tech Rep* 2005;10(4):186–8. <http://dx.doi.org/10.1016/j.jistr.2005.10.003>. URL, <http://dx.doi.org/10.1016/j.jistr.2005.10.003>.
- Law FY, Chan PP, Yiu S-M, Chow K-P, Kwan MY, Tse HK, et al. Protecting digital data privacy in computer forensic examination. In: *Systematic Approaches to Digital Forensic Engineering (SADFE)*, 2011 IEEE Sixth International Workshop on, IEEE; 2011. p. 1–6.
- Lee J, Un S. Digital forensics as a service: a case study of forensic indexed search. In: *ICT Convergence (ICTC)*, International Conference on, 2012; 2012. p. 499–503. <http://dx.doi.org/10.1109/ICTC.2012.6387185>.
- Madsen P, editor. Liberty Alliance project White paper: liberty ID-WSF people service – federated social identity; Dec. 2005.
- Microsoft Corporation. Outlook personal folders (.pst) file format. Tech. rep., Microsoft Corporation, [MS-PST] v20141019. 2014.
- Neuman BC, Ts'o T. Kerberos: an authentication service for computer networks. *Comm Mag* 1994;32(9):33–8. <http://dx.doi.org/10.1109/35.312841>. URL, <http://dx.doi.org/10.1109/35.312841>.
- Petitcolas FAP. Kerckhoffs' principle. In: van Tilborg HCA, Jajodia S, editors. *Encyclopedia of cryptography and security*. 2nd ed. Springer; 2011. p. 675. URL <http://dblp.uni-trier.de/db/reference/crypt/crypt2011.html#Petitcolas11>.
- Quick D, Choo K-KR. Impacts of increasing volume of digital forensic data: a survey and future research challenges. *Digit Investig* 2014;11(4): 273–94. <http://dx.doi.org/10.1016/j.diin.2014.09.002>. <http://www.sciencedirect.com/science/article/pii/S1742287614001066>. URL.
- Roussev V, Richard III GG. Breaking the performance wall: the case for distributed digital forensics. *Proceedings of the 2004 Digital Forensics Research Workshop*, Vol. 94; 2004.
- Roussev V, Quates C, Martell R. Real-time digital forensics and triage. *Digit Investig* 2013;10(2):158–67. URL, <http://dblp.uni-trier.de/db/journals/di/di10.html#RoussevQM13>.
- Shebaro B. Privacy-preserving techniques for computer and network forensics. Ph.D. thesis. 2012. Albuquerque, NM, USA, aAI3517591.
- Shvachko K, Kuang H, Radia S, Chansler R. The hadoop distributed file system. In: *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST'10. Washington, DC, USA: IEEE Computer Society; 2010. p. 1–10. URL, <http://dx.doi.org/10.1109/MSST.2010.5496972>.
- Song DX, Wagner D, Perrig A. Practical techniques for searches on encrypted data. In: *Security and Privacy*, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on, IEEE; 2000. p. 44–55.
- Tan YS, Ko RKL, Holmes G. Security and data accountability in distributed systems: a provenance survey. In: *Proceedings of the 15th IEEE International Conference on High Performance Computing and Communications (IEEE HPCC13)*. Zhang Jiajie, China: IEEE Computer Society; 2013.
- Tipton HF. Official (ISC)2 guide to the CISSP CBK. 2nd ed. Boston, MA, USA: Auerbach Publications; 2009.
- Ugen M. Scalable performance for a forensic database application. Master's thesis. University of Twente; April 2013. URL, <http://essay.utwente.nl/63429/>.
- U.S. Department of Justice, Office of Justice Programs, Office of Juvenile Justice and Delinquency Prevention. When your child is missing: a family survival guide. May 1998. URL, <http://www.ojjdp.gov/pubs/childsmismissing/>.
- van Baar B, van Beek H, van Eijk E. Digital forensics as a service: a game changer. *Digit Investig* 2014;11(Suppl. 1):S54–62. proceedings of the First Annual DFRWS Europe, <http://dx.doi.org/10.1016/j.diin.2014.03.007>, <http://www.sciencedirect.com/science/article/pii/S1742287614000127>. URL.
- Wen Y, Man X, Le K, Shi W. Forensics-as-a-service (FaaS): computer forensic workflow management and processing using cloud. In: *Cloud computing 2013, The Fourth International Conference on Cloud Computing, GRIDS, and Virtualization*; 2013. p. 208–14.
- Wikipedia. The first 48. URL, http://en.wikipedia.org/wiki/The_First_48.
- Yavuz AA, Ning P, Reiter MK. Baf and fi-baf: efficient and publicly verifiable cryptographic schemes for secure logging in resource-constrained systems. *ACM Trans Inf Syst Secur (TISSEC)* 2012; 15(2):9.
- Yavuz AA, Ning P, Reiter MK. Efficient, compromise resilient and append-only cryptographic schemes for secure audit logging. In: *Financial cryptography and data security*. Springer; 2012. p. 148–63.