Haute école de gestion de Genève
CRAG - Centre de Recherche Appliquée en Gestion
Cahier de recherche

Making Sense of Unstructured Memory Dumps from Cell Phones

David Billard, Rolf Hauri

Cahier : N° HES-SO/HEG-GE/C -- 09/2/1 -- CH

2009

Hes·SO GENÈVE
Haute Ecole Spécialisée de Suisse occidentale

Making Sense of Unstructured Memory Dumps from Cell Phones

David Billard
Rolf Hauri

Cahier de recherche

Avril 2009

Summary

This paper presents an alternative to traditional file carving, targeted to cell phone forensics. The proposed algorithm processes the cell phone memory dump thanks to a previous partial knowledge of the content of the regular files present in the memory dump. The memory dump is decomposed into elementary parts, each part classified according to the file type it is supposed to belong to, and finally ordered in a sequence representing the recovered file. The sequence is then transformed into a real file. This paper presents the first part of the algorithm (model and implementation) and does not cover the reordering of clusters nor the export of the recovered file. A reference to a basic open source software using this technology is provided.

Keywords

Forensics, memory dumps, file carving, cell phone.

Making Sense of Unstructured Memory Dumps from Cell Phones

David Billard, Rolf Hauri
University of Applied Sciences of Western Switzerland in Geneva
David.Billard@hesge.ch, Rolf.Hauri@hesge.ch

Abstract

This paper presents an alternative to traditional file carving, targeted to cell phone forensics. The proposed algorithm processes the cell phone memory dump thanks to a previous partial knowledge of the content of the regular files present in the memory dump. The memory dump is decomposed into elementary parts, each part classified according to the file type it is supposed to belong to, and finally ordered in a sequence representing the recovered file. The sequence is then transformed into a real file. This paper presents the first part of the algorithm (model and implementation) and does not cover the reordering of clusters nor the export of the recovered file. A reference to a basic open source software using this technology is provided.

Keywords: Forensics, memory dumps, file carving, cell phone.

1 Introduction

One of the goals of the digital forensic research community is to design methods and tools for extracting data from any kind of digital devices.

When undertaking a real investigation of a cell phone, forensic practitioners often use a combination of two approaches:

1. a logical extraction of all files present on the device;
2. a physical extraction of the memory device.

The first approach is done by using the cell phone manufacturer protocol to retrieve the regular files (not deleted) from the cell phone. This approach has many drawbacks since it does not preserve in a forensically sound way the integrity of the device, but it is sometimes the only approach done by a police officer.

The second approach is used by well equipped laboratories, with highly skilled engineers. It consists in desoldering the flash memory chip from the device and in reading its content via ad-hoc hardware. The reader can refer to [BdJK+07] and [Bre06] for a better understanding of the whole process. After the physical extraction is done, a long and painful process of understanding the structure of the file system begins. As a matter of fact, cell phones, and now netbooks with SSD, rely intensively on flash memory. Unfortunately for the forensic processing, due to the flash memory controller, the memory clusters are no longer stored in a contiguous manner and are disseminated on the whole memory. In addition, extra memory blocks, which are not part of the current state of the file system, are present in the memory dump[BdJK+07].

For that reason, the tedious analysis of the file system structure has for main objective a proper mapping of memory blocks into clusters. For the same reason, the main advantage of desoldering is to get a complete memory extraction, including the discarded memory blocks, and additional information concerning the usage of the blocks.

However, desoldering has severe drawbacks:

- it is an expensive technology (desoldering skills, dedicated hardware);
- it might change the state of the flash memory content if not properly done (overheating, for instance);

- it imposes the need to understand the file system structure and to write software to retrieve regular (non erased) logical files. This process may generate problems when the file system structure is unkown or hard to decode in a short time frame.

Once this process of desoldering, dumping the memory and understanding the file system structure is completed, the examiner has several data containers to consider:

- the set of logical files, extracted by using the file system structure;
- some hints of previous logical files that have been deleted, extracted by finding digital remnants inside the file system structure, for instance a directory entry flagged as deleted;
- the extra memory blocks that may contain information;
- the unallocated space where file carving can be experimented.

We claim that this result, achieved at great costs and skills, is not so different from the result obtained by using a logical dump of the cell phone flash memory combined with a logical extraction of all files present on the device.

The logical memory dump is an alternative to desoldering. It can be achieved by using flashboxes or dedicated software. Unfortunately, by using logical memory dumps, the examiner fails to retrieve the extra memory blocks, where some evidence might be stored.

If we consider this dual approach (logical dump of memory and logical extraction of regular files), there is no more need to reconstruct the file system structure. As a matter of fact, the examiner has enough data containers to consider:

- the set of logical files, extracted by using the logical extraction;
- the unallocated space where file carving can be experimented.

However, he does not have:

1. hints of previous logical files that have been deleted, extracted by finding digital remnants inside the file system structure, such as a directory entry marked as deleted, for example;
2. the extra memory blocks that may contain information.

The first point is a minor problem, except for dates, length and filename that could be stored in a deleted file entry. As a matter of fact, and at best, only the first cluster of a file can be retrieved, the other clusters being scattered around in the flash memory with few probability of being contiguous. Therefore this information does not give an important leverage on retrieving deleted files.

The second point is more problematic, and therefore a choice must be operated between using desoldering for extracting all the memory blocks where evidence can be stored, which is very costly, and extracting only the available memory blocks (via the controller), thus missing some blocks, but at a much lesser cost. This issue is particularly important in civil law countries, where the financial cost of a criminal investigation is almost always supported by the court of law.

After all the extractions have been done, no deleted file has been retrieved yet. This step will be done by analysing the unallocated space of the memory, which has no structure. Forensic practitioners often face the problem of extracting information from unstructured memory dumps: memory is then viewed only as a list of clusters.

Many tools, known as file carvers, try to reconstruct files by parsing the dump and finding file headers and footers, in the hope that the file content will be physically stored in contiguous clusters, from the header until the footer [Car06, Car05]. Unfortunately, due to the particular properties of flash memory at work in cell phones, this assumption does not hold and leads to bad results.

A major attempt in redesigning file carving can be found in [Coh07]. We use some results from this paper, in particular we suppose the *modulo rule* to be enforced, which means that files begin on sector boundaries and fragmentation can only occur on sector boundaries. We also propose a

use of some carving techniques from [Coh07] and [Gar07] in sections 5.3 and 5.4.

Our paper proposes an alternative to these tools based on previous partial knowledge of the content of regular files present in the memory dump. This technology still has many limitations when applied to disk memory dumps but shows some efficiency when applied to cell phones or digital cameras.

This paper focuses on the first part of the algorithm that strips the memory dump from useless clusters, organizes the remaining clusters by file types and provides recovered file skeletons. The second part of the algorithm, that reorders the clusters and exports the retrieved files is not covered in this paper. However, the first part of the algorithm gives some interesting results and we provide a basic open source software using this technology.

The paper is structured as follows. We present in section 2 the overall process needed to recover erased files, then we introduce our formal model and some definitions in section 3. Section 4 discusses some alternative choices in the model. In section 5 we present an experimental implementation of the formal model, using C language and databases.

2 Workflow

The final purpose of our algorithm is to correctly identify the deleted files or their fragments.

The first step of the workflow is to strip the memory dump from all the regular files. As a matter of fact, the regular files are known and can be studied at leisure. Besides, leaving these files in the dump would be a pollution for the future steps of the algorithm. The regular files represent the allocated space of the dump. We call this step the calculation of unallocated space. One may notice that the calculated unallocated space is also stripped of copies of parts of regular files that might be present in the real unallocated space, as a result of the system behavior (see section 4). Figure 1 shows an example of Step 1.

The second step identifies which possible files are still present in the unallocated space. To that purpose, we scan the clusters' content for headers and footers. The result of this step is a list of candidate files, based on their signature, and for each of these files, we have a first and last cluster of content. These first, *resp.* last clusters are called the originating, *resp.* finishing clusters. Figure 2 shows an example of Step 2 for the calculation of originating clusters.

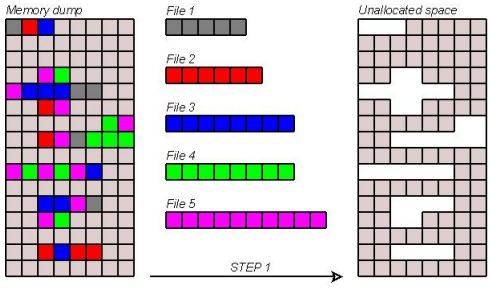

Figure 1: Calculation of unallocated space.

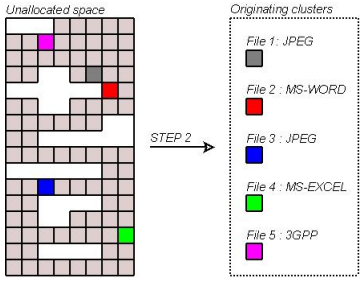

Figure 2: Calculation of originating clusters.

The third step is to apply heuristic methods in order to make a typology of clusters. We present one of these heuristics that merge steps 2 and 3, but many more sophisticated ones could be designed. The purpose of this step is to associate each cluster to a file type and to draw a chart of the unallocated space based on this information. Figure 3 shows an example of Step 3.

The fourth step, not covered in this paper, is to try different sequences among the clusters of the same typology, and to attach these sequences to the originating clusters. The sequences, prefixed with the originating clusters, represent a tentative deleted file. The sequences are built by calculating a distance from the originating clusters to the clusters of the same type. The relative position

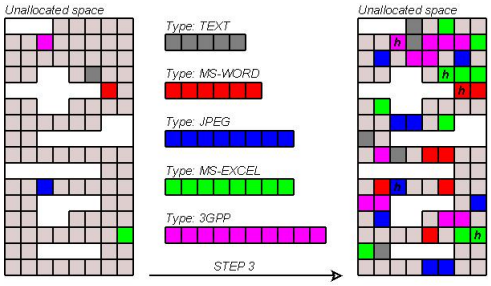

Figure 3: Calculation of clusters typology.

of the clusters, inside a sequence, will be defined either by their distance from the originating cluster of the sequence or by their distance from each other. Figure 4 shows an example of Step 4.

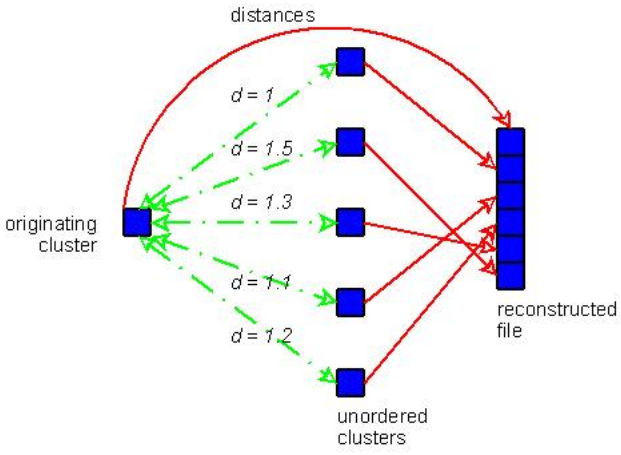

Figure 4: Calculation of a cluster sequence.

The remaining clusters, not yet considered by the previous steps, will require additional attention, in a non-automated way.

The final step is to run consistancy checks and statistics from the reconstructed files on the memory dump in order to determine if all files have been identified. Section 4 presents more information on this final step.

3 Formal Model and Definitions

In this section, we present the formal model of our approach and some definitions that will be used in section 5.

3.1 Step 1 - Calculation of Unallocated Space

We are using set arithmetics since the order of clusters is of no importance for our purpose. Furthermore, it helps in implementing the solution by using a relational database and SQL operations, optimized when dealing with sets. However, this calculation might lead to unwanted results, as presented in section 4.

Let EM be the content of the Extracted Memory (or Memory Dump).

$$EM = \{EmC_0, EmC_1, \cdots, EmC_n\}$$

Where EmC_i = Cluster i of EM. The size of a cluster is either known before hand, or a minimal value (for instance 512) can be used[1].

Let RFS be the set of regular files found on the device and extracted in a logical way.

$$RFS = \{Rf_0, Rf_1, \cdots, Rf_n\}$$

Where Rf_i = Regular file i. We suppose that the ending cluster of a regular file on the device has the same content as in the regular file on the investigation station[2]. Each RF_i is in turn composed of clusters:

$$Rf_i = \{Rf_iC_0, Rf_iC_1, \cdots, Rf_iC_n\}$$

Where Rf_iC_j = cluster j of regular file Rf_i.

From the previous set of definitions, the computation of unallocated space US becomes easy:

[1] The use of small values for cluster size might add great strain on the calculation of the cluster in step 4.

[2] Usually the last cluster of the file on the device is the same as the last cluster on the station. However, depending on the extraction method, it may occur that the trailing cluster of the file on the station is zeroed after the logical end of file.

$$US = EM - RFS$$

However, in order to practically achieve our algorithm, we need an efficient way to compare rapidly the content of two clusters, whatever their sizes. To that purpose, we use a hash function h that calculates the MD5 (or SHA-1, SHA-2) of any cluster.

Let $h(C_i)$ be the hash function h applied to cluster C_i and let hc_i be the resulting hash value. We define a function H, extending h to a set of clusters:

$$H(\{C_0, C_1, \cdots, C_n\}) = \{h(C_0), h(C_1), \cdots, h(C_n)\}$$

By transitivity of h, we obtain:

$$\{h(C_0), h(C_1), \cdots, h(C_n)\} = \{hc_0, hc_1, \cdots, hc_n\}$$

Therefore the computation of unallocated space $H(US)$ becomes:

$$H(US) = H(EM) - H(RFS)$$

3.2 Steps 2 & 3 - Calculation of Cluster Types

Some research has been done in order to predict the types of clusters. In [CC08], the use of linear discriminant and the use of longuest common substring and subsequences are analysed, whereas the use of distance from sample files is proposed in [KS06a] and [KS06b]. To a lesser extent, [LWSH05] proposes a file categorisation based on the statistical analysis of their binary content.

In this paper, we are using a pragmatic approach, not dissimilar from [Coh07].

The type of cluster is determined by some structural artifact found in the content of the cluster. For instance "\begin{itemize}" is a structural artefact of a LaTeX file, or "0xd0,0xcf,0x11,0xe0,0xa1,0xb1,0x1a,0xe1" is an artifact header for an MS-Office file.

In order to calculate a cluster type, we scan the cluster content against a set of artefacts. Each artefact is related to one (or more) file types and falls into three categories:

1. *headers*, if the artefact is found at the beginning of the file;
2. *footers*, if the artefact is found at the end of the file;
3. *artifacts*, in any other case.

These categories are represented as sets, which are used to classify the clusters.

When the cluster type is determined by a header, the cluster becomes an originating cluster:

$$C_i \in OriginatingClusters \rightarrow$$
$$\exists\, regex(C_i) \in\ headers$$

Where $regex(C_i)$ means a subpart of C_i. When the cluster type is determined by a footer, the cluster becomes an finishing cluster:

$$C_i \in FinishingClusters \rightarrow$$
$$\exists\, regex(C_i) \in\ footers$$

When the cluster type is determined by a structural artifact, but is neither an originating or finishing cluster, then the cluster becomes a structural artifact:

$$C_i \in StructuralArtefacts \rightarrow$$
$$(\exists\, regex(C_i) \in artifacts)\ AND$$
$$(C_i\ \notin\ (OriginatingClusters \cup$$
$$FinishingClusters))$$

This crude division between *OriginatingClusters*, *FinishingClusters* and *StructuralArtefacts* is a very primitive heuristic for classifying clusters. In case of unclassified clusters or a cluster with too few artefacts to rely on, the cluster becomes an unclassified cluster:

$$C_i \in UnclassifiedCluster \rightarrow$$

$$C_i \notin (OriginatingClusters \cup FinishingClusters \cup StructuralArtefacts)$$

3.3 Step 4 - Sequencing Clusters

This step is not fully covered by this paper. The closest work concerning cluster sequencing can be found in [C.J07] which is intended for file type differenciation but could be used with profit for sequencing. Basically, our approach uses distance calculation among clusters, taking as hypothesis that for the same file ef_i, two logical contiguous clusters share $x\%$ common bytes. From this hypothesis, tentative clusters sequences S_k are drawn from one originating cluster and tried.

Let EF be the set of recovered files:

$$EF = \{ef_0, ef_1, \ldots, ef_n\}$$

Where ef_i is the recovered file i.

$$ef_i = (\{OriginatingCluster_i\}, S_i, \{FinishingCluster_i\})$$

Where S_i represents the "most plausible tentative sequence" of clusters for ef_i.

This algorithm is augmented with the knowledge we might have from specific file types. For instance some structures are found before others in the normal sequencing of a file.

4 Unwanted results

4.1 Duplicated Files

In a file system, it may happen that the same file is found several times, either as a regular or erased file. With the increase of memory size, this is not an uncommon thing. By relying on cluster content, our algorithm will retain only one copy for each identical cluster, corresponding to a regular file.

The drawback of this behavior is the impossibility to collect statistics on the occurences of a regular file, either in non-deleted or deleted form. However, it is always possible to count how many times each cluster from this file is found in the memory dump.

4.2 Files with some clusters having identical content

Another kind of undesired behavior of the algorithm is when different files have identical clusters with the same content. For instance, video files with the same first minutes, text files with identical sections, etc. Our algorithm retaining only one copy for each identical cluster, this might lead in holes in files, or inconsistent files, after step 4. One way to deal with this problem is to retain each copy of identical clusters. However it will add more complexity in steps 3 and 4. Another way to handle this problem is to manually reconsider inconsistent retrieved files and to fetch missing clusters in the memory dump.

4.3 Identical cluster contents not associated to a file

In a file system, some clusters may never have been used. These clusters are usually left blank (in flash memory, all bytes are set to 0xFF). Left unchanged, our algorithm will consider all these clusters as valid clusters. We can overcome this problem by creating a virtual "evidence", composed of only one cluster filled with 0XFF. Threfore, our algorithm will retain only one cluster instead of each copy, thus reducing the number of candidate clusters for step 2. Unfortunately, some files may have blanked clusters, because the data in this cluster is composed only of 0xFF. In that case, we are back to the scenario of section 4.2.

5 Implementation

This section presents the implementation of the previous model. The implementation has been published on the open source platform `http://www.sourceforge.net`, under the project "Forensic File Carving Tools (forensicfct)".

Our algorithm extensively uses set computation and we naturally choose a database management

system for its persistent storage facility and also its highly optimized query engine. The major drawback of using a DBMS is the heavy load of the transactional system. However, we believe that the DBMS is much more flexible than traditional file systems when dealing with forensic operations.

We used the PostgreSQL DBMS system, the libpq library, C language and an operating system (Linux or POSIX) allowing direct file mapping into memory.

5.1 Database Model

The database model we designed is represented in figure 5.

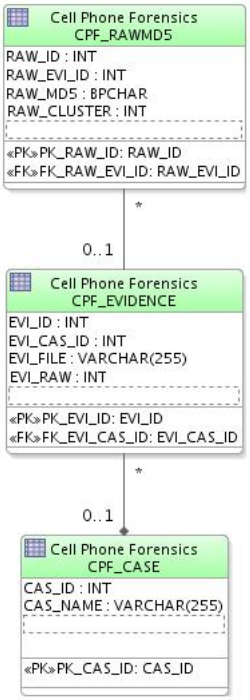

Figure 5: Database Model.

- The *CPF_CASE* table is used to manage case information. A very minimal set of information is kept there, essentially *CAS_NAME*, the name of the case.
- The *CPF_EVIDENCE* table is used to manage evidence files. Note that at the implementation level, no difference is made between the dump image to be analysed and the regular files present in the file system. Each evidence file is related to a case via the *EVI_CAS_ID* foreign key. *EVI_FILE* is the file name in the investigator's analysis station file system. *EVI_RAW* is set to 0 if the evidence file is the memory dump and *EVI_RAW* is set to 1 if the evidence file is a regular file.
- The *CPF_RAWMD5* table holds the value of the hash function applied to each cluster. Each cluster is related to an evidence file via the *RAW_EVI_ID* foreign key. *RAW_MD5* is the hash value for the cluster *RAW_CLUSTER*.

5.2 Step 1 - Calculation of Unallocated Space

The file containing the memory dump EM is mapped to the memory using *mmap*. Each cluster content[3] is hashed and the resulting hash value is stored in the *CPF_RAWMD5* table.

The regular logical files RF extracted from the cell phone are in turn mapped to the memory. Each cluster content of a rf_i file is hashed and the resulting hash value is stored in the *CPF_RAWMD5* table.

Upon completion of these two actions, the tables *CPF_CASE*, *CPF_EVIDENCE* and *CPF_RAWMD5* are fully loaded with the evidence. In order to calculate the unallocated space, we compute the $H(EM)$ and $H(RFS)$. These two sets can be represented as views on *CPF_RAWMD5*, or as temporary tables, or as standard tables. For the sake of clarity of the program, we choose to create new tables, of the same structure as *CPF_RAW_IMAGE*, and with names prefixed by the case number.

We create the table *CPF_RAW_IMAGE* that holds only the $H(EM)$. The query for the construction[4] of $H(EM)$ is represented in table 1.

We create the table *CPF_REG_IMAGE* that holds only the $H(RFS)$. The query for the

[3]The size of a cluster is stored in a constant.

[4]For the sake of clarity, we omit to indicate the case id and the mention of the database schema, for all the queries.

```
CREATE TABLE CPF_RAW_IMAGE WITH OIDS AS
    SELECT RAW_ID, RAW_EVI_ID, RAW_MD5, RAW_CLUSTER
    FROM CPF_EVIDENCE
    JOIN CPF_RAWMD5 ON EVI_ID=RAW_EVI_ID
    WHERE EVI_RAW = 1;
```

Table 1: Query for the construction of $H(EM)$

construction of $H(RFS)$ is represented in table 2. This query only differs from the construction of *CPF_RAW_IMAGE* by the restriction on *EVI_RAW*.

```
CREATE TABLE CPF_REG_IMAGE WITH OIDS AS
    SELECT RAW_ID, RAW_EVI_ID, RAW_MD5, RAW_CLUSTER
    FROM CPF_EVIDENCE
    JOIN CPF_RAWMD5 ON EVI_ID=RAW_EVI_ID
    WHERE EVI_RAW = 0;
```

Table 2: Query for the construction of $H(RFS)$

In order to create the table that will hold $H(US) = H(EM) - H(RFS)$, we use the *EXCEPT* statement (or *MINUS* in Oracle) that takes every row from a table that is not present in the second table. The query for the construction of $H(US)$ is represented in table 3. Note that we retain the *RAW_ID*, necessary to retrieve the content of the cluster back in the memory dump.

```
CREATE TABLE CPF_DIFF_IMAGE WITH OIDS AS
    (SELECT RAW_ID, RAW_EVI_ID, RAW_MD5, RAW_CLUSTER
    FROM CPF_RAW_IMAGE)
    EXCEPT
    (SELECT RAW_ID, RAW_EVI_ID, RAW_MD5, RAW_CLUSTER
    FROM CPF_REG_IMAGE);
```

Table 3: Query for the construction of $H(US)$

5.3 Steps 2 & 3 - Calculation of Cluster Types

Some tools already exist to find headers in binary streams. These tools can be modified to find headers, footers and structural artifacts of files to be retrieved. Moreover, for some known file types, tools like Hachoir can extract metadata contained in the first cluster of a file. For the purpose of this paper, we used a basic set of functions:

- *getHeader()* – returns the header id, or 0 if no header;
- *getFooter()* – returns the footer id, or 0 if no footer;
- *getArtefact()* – returns the artifact id, or 0 if no artifact.

The file containing the memory dump *EM* is mapped to the memory using *mmap*. Each cluster is checked against *getHeader()*, *getFooter()* and *getArtefact()*. These functions use a table *CPF_FILETYPE*, giving the list of file types and a table *CPF_ARTIFACT* defining the list of artifact for each table. These tables are represented in figure 6.

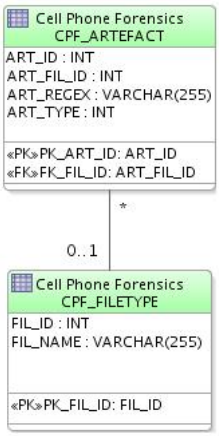

Figure 6: Tables CPF_FILETYPE and CPF_ARTIFACT.

The output of these functions is stored in a third table, *CPF_CLUSTER*, which gives for each cluster its type (according to the file type in *CPF_FILETYPE*), and its category in {unknown, header, footer, artefact}.

The table *CPF_CLUSTER* is a chart of the unallocated space and can be searched by file type and category, which drammatically reduces the complexity of file reconstruction. The final database model is represented in figure 7.

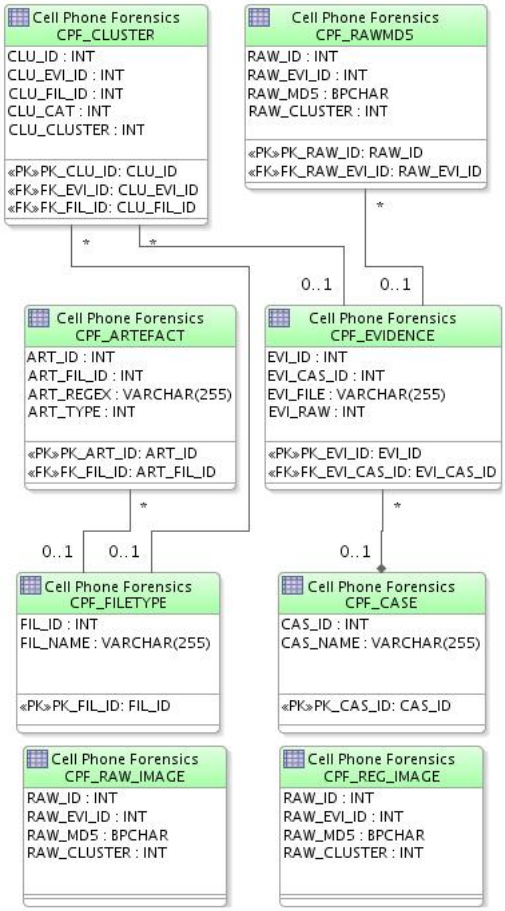

Figure 7: Whole architecture.

5.4 Step 4 - Sequencing Clusters

In order to sequence the clusters, we add a new table *CPF_CANDIDATES*. Each row in this table will represent a possibility of file. For each cluster of category *header*, we add rows of clusters from he same file type, with their relative distance from the first cluster. The calculation of the distance is not covered by this paper, and the calculation might differ depending on the file type.

5.5 Extraction of Recovered Files

The file containing the memory dump EM is mapped to the memory using *mmap*. The extraction of the recovered files is done by taking in turn each tentative file, from *CPF_CANDIDATES*, retrieving in sequence each data block associated to the cluster in EM and writing them to a physical file on the analysis workstation.

5.6 Besides File Recovery

In addition to file recovery, our database scheme can be extremely usefull when looking for remnants of a known file inside an arbitrary memory dump. For instance, the authors have been given a video file V and have been asked to answer this question: "Has video file V been played on this computer?".

By filling the memory dump of the device and the content of V in our database (step 1 of the algorithm), the answer to this question is as simple as:

```
SELECT * FROM
    (SELECT RAW_ID, RAW_EVI_ID, RAW_MD5, RAW_CLUSTER
    FROM CPF_RAW_IMAGE)
    INTERSECT
    (SELECT RAW_ID, RAW_EVI_ID, RAW_MD5, RAW_CLUSTER
    FROM CPF_REG_IMAGE);
```

Table 4: Query for finding a remnant of V

This type of question becomes increasingly relevant in criminal cases, when digital evidence has been erased at a distant time in the past and the digital device has been used in the meantime.

5.7 Comments and Code Repository

The source code is available under the BSD licence at: `https://sourceforge.net/projects/forensicfct/`

It is provided with a working example: a cell phone memory dump and the logical files extracted from the phone.

6 Conclusion

In this paper we have shown that file recovery in cell phone forensics can be achieved via logi-

cal memory dumps and file extraction instead of heavy methods involving desoldering and in-depth study of file system structure. The method we devised is slightly less effective in terms of raw data retrieval, but much more flexible and cheaper. The method is based on set arithmetics and is implemented using C, databases and SQL queries.

The algorithm calculates the unallocated space of a raw digital device, identifies candidate files based on an originating cluster and subsequently associates clusters to file types. The clusters inside a file type are sequenced and give files as an output.

Futhermore, our method can be extremely useful besides file recovering, when cluster comparison is of primer importance, for instance when verifying the past presence of known files in a digital device.

7 Future works

This method needs to be improved in many ways. First of all, the method allowing the correct association of clusters to file type can take more avantage of statistical methods as in [KS06a, KS06b, LWSH05, C.J07]. Furthermore, the algorithms in [C.J07] can be adapted, not only for cluster classification, but also for their ordering. On the implementation level, a great effort has to be made to make the program user-friendly and optimized.

Finally, we plan to make a comparison of the same cell phone investigated by using desoldering with file system structure decoding and by running our algorithm.

References

[BdJK+07] Marcel Breeuwsma, Martien de Jongh, Coert Klaver, Ronald van der Knijff, and Mark Roeloffs. Forensic data recovery from flash memory. *Small Scale Digital Device Forensics*, 1(1):1 – 17, 2007.

[Bre06] Ing. M.F. Breeuwsma. Forensic imaging of embedded systems using jtag (boundary-scan). *Digital Investigation*, 3(1):32 – 42, 2006.

[Car05] Brian Carrier. *File System Forensic Analysis*. Addison-Wesley Professional, 2005.

[Car06] Brian D. Carrier. *A hypothesis-based approach to digital forensic investigations*. PhD thesis, West Lafayette, IN, USA, 2006. Adviser-Spafford,, Eugene H.

[CC08] William C. Calhoun and Drue Coles. Predicting the types of file fragments. *Digital Investigation*, 5(Supplement 1):S14 – S20, 2008. The Proceedings of the Eighth Annual DFRWS Conference.

[C.J07] Veenman C.J. Statistical disk cluster classification for file carving. *Third International Symposium on Information Assurance and Security*, pages 393–398, August 2007.

[Coh07] M.I. Cohen. Advanced carving techniques. *Digital Investigation*, 4(3-4):119 – 128, 2007.

[Gar07] Simson L. Garfinkel. Carving contiguous and fragmented files with fast object validation. *Digital Investigation*, 4(Supplement 1):2 – 12, 2007.

[KS06a] M. Karresand and N. Shahmehri. File type identification of data fragments by their binary structure. *Information Assurance Workshop, 2006 IEEE*, pages 140–147, June 2006.

[KS06b] Martin Karresandn and Nahid Shahmehri. *Oscar File Type Identification of Binary Data in Disk Clusters and RAM Pages*, pages 413–424. IFIP International Federation for Information Processing. Springer Boston, 2006.

[LWSH05] Wei-Jen Li, Ke Wang, S.J. Stolfo, and B. Herzog. Fileprints: identifying file types by n-gram analysis. *Information Assurance Workshop, 2005. IAW '05. Proceedings from the Sixth Annual IEEE SMC*, pages 64–71, June 2005.

Cahiers de recherche du Centre de Recherche Appliquée en Gestion (CRAG) de la Haute Ecole de Gestion - Genève

CRAG – Centre de Recherche Appliquée en Gestion
Haute école de gestion - Genève
Campus de Battelle, Bâtiment F
7, route de Drize – 1227 Carouge – Suisse
✉ crag@hesge.ch
www.hesge.ch/heg/crag
✆ +41 22 388 18 18
🖷 +41 22 388 17 40

2006

- N° HES-SO/HEG-GE/C--06/1/1--CH
  Andrea BARANZINI
  Damien ROCHETTE
  "La demande de récréation pour un parc naturel. Une application au Bois de Pfyn-Finges, Suisse"

- N° HES-SO/HEG-GE/C--06/2/1--CH
  Giovanni FERRO LUZZI
  Yves FLÜCKIGER
  Sylvain WEBER
  "A Cluster Analysis of Multidimentional Poverty in Switzerland"

- N° HES-SO/HEG-GE/C--06/3/1--CH
  Giovanni FERRO LUZZI
  Sylvain WEBER
  "Measuring the Performance of Microfinance Institutions"

- N° HES-SO/HEG-GE/C--06/4/1--CH
  Jennifer D'URSO
  "L'eau de boisson : Aspects logistiques et attitude du consommateur"

- N° HES-SO/HEG-GE/C--06/5/1--CH
  Jennifer D'URSO
  "La gestion publique de l'eau en Suisse"

- N° HES-SO/HEG-GE/C--06/6/1--CH
  Philippe THALMANN
  Andrea BARANZINI
  "Gradual Introduction of Coercive Instruments in Climate Policy"

- N° HES-SO/HEG-GE/C--06/7/1--CH
  Andrea BARANZINI
  Caroline SCHAERER
  José RAMIREZ

Philippe THALMANN
"Feel it or Measure it. Perceived vs. Measured Noise in Hedonic Models"

- N° HES-SO/HEG-GE/C--06/8/1--CH
  José RAMIREZ
  Anatoli VASSILIEV
  "An Efficiency Comparison of Regional Employment Offices Operating under Different Exogenous Conditions"

- N° HES-SO/HEG-GE/C--06/9/1--CH
  José RAMIREZ
  Joseph DEUTSCH
  Yves FLÜCKIGER
  Jacques SILBER
  "Export Activity and Wage Dispersion : The Case of Swiss Firms"

- N° HES-SO/HEG-GE/C--06/10/1--CH
  Joëlle DEBELY
  Gaëtan DERACHE
  Emmanuel FRAGNIERE
  Jean TUBEROSA
  "Rapport d'enquête : sondage Infobésité"

- N° HES-SO/HEG-GE/C--06/11/1--CH
  Andrea BARANZINI
  José RAMIREZ
  Cristian UGARTE ROMERO
  "Les déterminants du choix de (dé)localisation des entreprises en Suisse"

- N° HES-SO/HEG-GE/C--06/12/1--CH
  Catherine EQUEY BALZLI
  Jean TUBEROSA
  David MARADAN
  Marie-Eve ZUFFEREY BERSIER
  "Étude du comportement des PME/PMI suisses en matière d'adoption de système de gestion intégré. Entre méconnaissance et satisfaction."

- N° HES-SO/HEG-GE/C--06/13/1—CH
  Joëlle DEBELY
  Magali DUBOSSON
  Emmanuel FRAGNIÈRE
  "The pricing of the knowledge-based services : Insight from the environmental sciences"

2007

- N° HES-SO/HEG-GE/C--07/1/1--CH
  Andrea BARANZINI
  Caroline SCHAERER
  "A Sight for Sore Eyes
  Assessing the value of view and landscape use on the housing market"

- N° HES-SO/HEG-GE/C--07/2/1--CH
  Joëlle DEBELY
  Magali DUBOSSON
  Emmanuel FRAGNIÈRE
  "The Travel Agent: Delivering More Value by Becoming an Operational Risk Manager"

- N° HES-SO/HEG-GE/C--07/3/1--CH
  Joëlle DEBELY
  Magali DUBOSSON
  Emmanuel FRAGNIÈRE
  "The Consequences of Information Overload in Knowledge Based Service Economies"

- N° HES-SO/HEG-GE/C--07/4/1--CH
  Lucie BEGIN
  Jacqueline DESCHAMPS
  Hélène MADINIER
  "Une approche interdisciplinaire de l'intelligence économique"

- N° HES-SO/HEG-GE/C--07/5/1--CH
  Journée de la recherche HEG 2007
  "Recueil des communications"

- N° HES-SO/HEG-GE/C--07/6/1--CH
  Sylvain WEBER
  Andrea BARANZINI
  Emmanuel FRAGNIÈRE
  "Consumers Choices among Alternative Electricity Programs in Geneva – An Empirical Analysis"

2008

- N° HES-SO/HEG-GE/C--08/1/1--CH
  Andrea BARANZINI
  José RAMIREZ
  Sylvain WEBER
  "The Demand for Football in Switzerland : An Empirical Estimation"
- N° HES-SO/HEG-GE/C--08/2/1--CH
  Giuseppe CATENAZZO
  Gaëtan DERACHE
  Emmanuel FRAGNIÈRE
  Patricia HUGENTOBLER
  Jean TUBEROSA
  "Rapport d'enquête préliminaire : Dessine-moi un service ! Entreprises et administration : comment concevoir et valoriser vos services"

- N° HES-SO/HEG-GE/C--08/3/1--CH
  Nguyen VI CAO
  Emmanuel FRAGNIÈRE
  Jacques-Antoine GAUTHIER
  Marlène SAPIN
  Eric WIDMER
  "Optimizing the marriage market through the reallocation of partners : An application of the linear assignment model"

- N° HES-SO/HEG-GE/C--08/4/1--CH
  Magali DUBOSSON
  Emmanuel FRAGNIÈRE
  Bernard MILLIET
  "A Control System Designed to Address the Intangible Nature of Service Risks"

- N° HES-SO/HEG-GE/C--08/5/1--CH
  Giuseppe CATENAZZO
  Jennifer D'URSO
  Emmanuel FRAGNIÈRE
  Jean TUBEROSA
  "Influences of Public Ecological Awareness and Price on Potable Water Consumption in the Geneva Area"

- N° HES-SO/HEG-GE/C--08/6/1--CH
  Alexandra BROILLET
  Magali DUBOSSON
  "Analyzing Web 2.0 Internet users in order to drive innovation in distribution strategy of luxury watches : A netnography analysis"

- N° HES-SO/HEG-GE/C--08/7/2--CH
  Alexandra BROILLET
  Magali DUBOSSON
  "Luxury e-services at the pre- and after-sales stages of the decision making process: Watch, car, art and travel blogs analysis"

- N° HES-SO/HEG-GE/C--08/8/1--CH
  Nicolas BUGNON
  René SCHEIDER
  "OPACs et utilisateurs - L'étude ACUEIL démontre les comportements de recherche et propose des outils simplifiés et flexibles"

- N° HES-SO/HEG-GE/C--08/9/1--CH
  Giuseppe CATENAZZO
  Emmanuel FRAGNIÈRE
  "Attitudes Regarding New Enterprise Risk and Control Regulations by the Active Population of the Geneva Area"

- N° HES-SO/HEG-GE/C--08/10/1--CH
  Giuseppe CATENAZZO
  Emmanuel FRAGNIÈRE
  "Identifying Bank Runs Signals through Sociological Factors: An Empirical Research in the Geneva Area"

- N° HES-SO/HEG-GE/C--08/11/1--CH
  Caroline SCHAERER
  Andrea BARANZINI
  "Where and How Do Swiss and Foreigners Live? Segregation in the Geneva and Zurich Housing Markets"

- N° HES-SO/HEG-GE/C--08/12/1—CH
  Giuseppe CATENAZZO
  Jennifer D'URSO
  Emmanuel FRAGNIÈRE
  "Elements of perception regarding sustainable development in Geneva"

- *N° HES-SO/HEG-GE/C--08/13/1--CH*
  Emmanuel FRAGNIÈRE
  Nils S. TUCHSCHMID
  Qun ZHANG
  "Liquidity Adjusted VaR Model: An Extension"

2009

- *N°HES-SO/HEG-GE/C—09/1/1—CH*
  Tobias Muller
  José Ramirez
  « Wage inequality and segregation between native and immigrant workers in Switzerland : evidence using matched employee-employer data »