Memory Carving in Embedded Devices: Separate the Wheat from the Chaff

Thomas Gougeon, Morgan Barbier, Patrick Lacharme, Gildas Avoine, Christophe Rosenberger

Memory carving in embedded devices: separate the wheat from the chaff

Thomas Gougeon[1], Morgan Barbier[1], Patrick Lacharme[1], Gildas Avoine[2,3], and Christophe Rosenberger[1]

[1] Normandie Universite; ENSICAEN-UNICAEN-CNRS, GREYC UMR 6072, F-14032 Caen, France
[2] INSA Rennes, IRISA UMR 6074
[3] Institut Universitaire de France

Abstract. Embedded devices usually gather and store personal data about the behaviours of their holders. For example, a public transportation card may record the last trips of the passenger, or a car ignition key may store the fuel consumption and the average engine speed of the vehicle. Being able to interpret these raw data without the knowledge of the specifications can be useful to establish digital evidence, for example in connection with criminal investigations.
This paper investigates memory carving techniques for embedded devices. Given that cryptographic material in memory dumps makes carving techniques inefficient, we introduce a methodology to distinguish meaningful information from cryptographic material in small-sized memory dumps. The proposed methodology uses an adaptive boosting technique with statistical tests. Experimented on EMV cards, the methodology reaches a successful recognition rate greater than 99.8%.

Keywords: forensics, memory carving, randomness, embedded devices, smartcards, privacy.

1 Introduction

Embedded devices usually gather and store personal data about the behaviours of their holders. They are typically low-cost devices including (but not limited to) credit cards, mass transportation passes, electronic passports, keyless entry and start systems, and ski passes. They usually gather and store a lot of personal data, for example an electronic passport contains the identity and the picture of its holder [3], a mass transportation pass may store the last trips of its holder [5], a ski pass may also contain the location of the ski lifts the skier used [1], an EMV card records the last payments done by the customer [9], a car ignition key in recent cars contains plenty of information about the car and the behaviour of the driver, including the monthly fuel consumption, the external temperature during the last trip, and the average engine speed. In most cases, the personal data contained in these devices are accessible without requiring any authentication,

and can be obtained using, for example, the ISO/IEC 7816 interface or by sniffing a genuine communication between the device and a reader.

Interpreting the meaning of the captured raw data is hard when the system specifications are not available. However, such a task is important today when investigations must be carried out. It can be to find digital evidence for example in connection with criminal investigations – when information related to a suspect is stored in a device – or to verify that a system complies with the national privacy regulations.

A large body of literature exists in the field of memory forensics. Many off-the-shelf tools exist, too. The analyses typically focus on hard drives [19] and volatile memories [4]. Analyses of hard drives are typically based on file carving, i.e., a technique that consists in searching for files in the considered data. The main difficulty is the file fragmentation in the system. File carving is consequently performed using machine learning techniques, the entropy of the blocks, or the file headers and footers. The technique targets specific file formats, e.g., PDF, ZIP [6], or file systems such as NTFS [26]. Analyses of volatile memories consist in searching for special strings or signatures, interpreting internal kernel structures, or enumerating and correlating all page frames, in order to retrieve running and terminated processes, open ports, sockets, hidden data, etc.

The analysis of the non-volatile memory of an embedded device differs from classical memory forensics techniques for several reasons. (i) First of all, the memory typically consists of a few kilobits only. (ii) The data available in these devices are poorly structured: in most cases, there are no file headers, sentinels, or field separators. (iii) Home-made encoding systems are commonly used in practice to save memory or to naively hide information. (iv) Performing a bit-by-bit copy of the memory is rarely possible because the only way to access the memory is to use the application program interface (API) or to eavesdrop a genuine communication. This means that the captured data is not necessarily a perfect copy of the memory.

A naive technique to interpret data retrieved from embedded devices consists in applying several encoding functions on the dumps until retrieving the correct one for each information stored. Due to the nature of the dumps, there is unfortunately no oracle that can efficiently determine whether the decoding of the information is correct. As a consequence, the technique outputs many false positives that renders it unusable in practice. Most existing contributions on the memory carving problem for embedded devices consider ad-hoc, hand-made analyses, e.g., for retrieving keys hidden in an EEPROM [7].

There exist few techniques designed for an automatic analysis of embedded devices. A seminal work, though, is due to Ton Van Deursen et al. [25], who investigated the memory carving problem for sets of memory dumps, and applied it to public transportation cards. It is worth noting that they obtained the memory dumps using the API of the cards, meaning that there is no guarantee that the dumps are indeed bit-by-bit copy of the memory. The authors aimed to singulate the memory data fields using the concept of commonalities and dissimilarities applied to a dump set. A commonality occurs for a given bit position

if the value of the bit is the same for all the dumps of a given set, whereas a dissimilarity occurs otherwise. Using these commonalities and dissimilarities, as well as contextual information (as data printed on the coupon), the technique deduces the data fields. Once the data fields are singulated, a manual investigation is needed to retrieve the encoding function. The authors applied their technique on the E-Go System (the public transportation card in Luxembourg) and retrieved a dozen of fields, e.g., the date and time of the last validation. Their work does not provide an automatic interpretation of the data and it requires contextual information to complete the analysis. Another work related to ours is due to Jean-Louis Lanet et al. [15], who investigated the reverse engineering of EEPROM area of Java Cards, retrieving the location of the Java Card source code and data (package, class, instance ...) related to the Java Card. Source code is located computing the index of coincidence (IC) [12] as for the cryptanalysis of the Vigenère cipher, whereas the data is found using pattern matching on headers (or metadata) that differ for each data type. In our case, the number of bits used by the encoding function of each information is variable (and not known), and each information seem to be stored on too few bits to look at the repetition of patterns, that complicate the use of the IC. Concerning the pattern matching, as said before, there is no header or metadata in our dumps.

Given the difficulty to retrieve personal data from the memory dump of an embedded device, this work focuses on a narrower problem that consists in distinguishing meaningful information (encoded with ASCII, BCD, etc.) from cryptographic material (ciphered data, hash value, secret key, etc.). The rationale behind this restriction is that cryptographic materials generate many false positives and no personal information can be obtained from these values, assuming the algorithms used to create the materials are cryptographically secure. As a consequence, we introduce a technique that *separates the wheat from the chaff*, namely a preliminary step in the forensics process that distinguishes meaningful information from cryptographic materials, considered as random data. Unfortunately, the size of the considered dumps does not allow to use classical tools (e.g., NIST's statistical tests [20]) that usually require several kilobytes of data to make the statistical tests relevant. Moreover, the tests cannot be naively applied to the data because the considered dumps contain data fields, which must be analysed separately. For the same reason, techniques for locating cryptographic keys hidden in gigabytes of sparse data, proposed by [22] and based on the entropy computation, are not possible on such dumps.

This paper introduces a statistical and automatic recognition technique that distinguished meaningful information from cryptographic material, obtained from non-volatile memory dumps of embedded devices. The technique, based on a machine learning method, called *boosting* [10], requires information neither on the dump structure, nor on the application context, for the classification between these two sets of data. The technique performs a differential analysis: comparing dumps of different devices belonging to the same application. Our technique reaches quite a high success rate: we applied it on EMV-based dumps and Calypso-based dumps, obtaining a 99% success rate.

2 Dump examples

To illustrate the problem considered in this paper, Section 2 provides details on two dumps extracted from EMV and Calypso cards. The cards contain elementary files that have been retrieved using the cards' APIs. The files are made of **records**. Files can be *linear fixed* (linear data structure of fixed length), *linear variable* (linear data structure of variable length), or *cyclic* (oldest data are erased to store newest data). The information is contained in (non-necessarily contiguous) **fields**, e.g., holder's name, holder's zip code, a cryptographic key, etc. A pedestrian approach has been used to analyse the dumps, given that there does not exist automatic tools that can achieve this task.

2.1 EMV dump

Figure 1 is a (partial) anonymised dump of a credit card compliant with the EMV specifications [9]. A binary representation of the data is used because the fields are not byte-aligned. The four highlighted and numbered sequences are fields that contain the holder name, the issuer's public key modulus of the card, the amount and the date of one of the last transactions.

Fig. 1: Extract of an anonymised credit card dump.

Holder name. The yellow bit sequence (or sequence 1) represents the name of the holder of the credit card (*MR James Smith*) encoded using ASCII and padded with the repeated pattern 0x20.

Issuer's public key modulus. The green sequence (or sequence 2) represents the issuer public key modulus used by the authentication protocol.

Transaction. The sequences highlighted in grey (sequence 3) and red (sequence 4) come from the same transaction (cash withdrawal). The grey sequence represents the transaction amount (230.00 euros) and the red one is the date of the transaction (2010/05/02).

The EMV card also contains a cyclic file that stores information on the transactions. For any new transaction, the information in the cyclic file is rotated such that the record about the oldest transaction is discarded to save room for the newest transaction.

2.2 Calypso dump

Figure 2 is a (partial) anonymised dump of a transportation card compliant with the Calypso specifications [5]. The record names (`ICC`, `Holder1`, `Holder2`, etc.) are available in the specifications, but the content of the records is not defined by Calypso. The content is indeed let to the discretion of the public transportation operator. The provided example illustrates that a single card may contain several encoding rules, and the information in the card is not necessarily byte-aligned.

```
ICC: 00 00 00 00 00 00 00 04 00 71 B3 00 00 00 00 00 01 B8 C8 21 02 50 00 33 01 1A 13 45 00
Holder1: 04 00 98 E5 94 C8 02 0D C4 C1 E0 5B D5 90 00 00 00 00 00 00 00 19 84 10 23 52 82 D2 CF
Holder2: F3 6A 68 88 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
EnvHol1: 08 38 2B 00 08 BD 59 2A 46 5D 02 49 98 E5 94 C8 02 0D C4 C1 E0 59 41 F4 00 00 00 00
EnvHol2: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
EvLog1: 09 10 85 92 04 20 60 86 60 00 00 00 0F 8D E0 66 F6 40 00 05 80 00 00 51 08 57 A1 1B A0
EvLog2: 09 10 85 7A 04 20 10 86 61 1B A0 0B 0F 82 90 E4 D6 40 00 05 40 00 00 11 08 57 A1 1B A0
EvLog3: 09 10 85 5A 04 20 60 86 61 17 60 0B 0F 82 8C 02 D6 40 00 05 00 00 00 91 08 50 21 17 80
```

Fig. 2: Anonymised transportation car dump.

On the `Holder1` line, the first frame represents the BCD-encoded birth date of the holder: 1984/10/23. The second frame that continue on the `Holder2` line represents the name of the holder, namely "James Smith". To decode this information the binary representation of the sequence must be split into 5-bit pieces (omitting the first bit of the sequence), which are then decoded with the rule (decimal representation): `A=1`, `B=2`, `C=3`, etc.

`EvLog1`, `EvLog2`, and `EvLog3` are the last three trips performed by the card, stored in a cyclic file. For example, the first frame in the `EvLog1` line corresponds to the validation time, which is here "11:53 am". This information is retrieved using the binary representation of 0x592 (omitting the last bit), and converting it to an integer that represents the number of minutes since the beginning of the day. The second frame represents the validation date of the card during the trip: 2008/12/09. This information is retrieved by using the binary representation of 0x5108 (omitting the two first bits) and by converting it to an integer that represents the number of days since 1997/01/01. Other information on this line are the transportation means (metro, bus, tramway), the bus line number, the number of travellers who shared the card for that trip, the station, etc. Additional information can be found in the dump, e.g., the serial number of the card, the manufacturer, the date of manufacture, etc.

3 Statistical analysis

Retrieving the meaningful information from a dump using a statistical analysis is a difficult problem. In particular, the meaningful information is drowned in a mass of information that include pseudo-random values generated by cryptographic means. This paper consequently focuses on a preliminary step in the forensics process that distinguishes meaningful information from cryptographic materials. To start with, we explain below the difficulty to use statistical tests to perform this task in our framework.

3.1 Statistical tests for (pseudo-) random generators

There exist many statistical tests for random and pseudo-random number generators. The NIST statistical test suite includes the most important ones, while keeping small the redundancy between them. We consequently decided to consider this suite for our experiments.

A statistical test aims to verify a given *null hypothesis*, which is *data are random* in our experiments. A p-value represents the strength of the evidence against the null hypothesis. This p-value is computed from the reference distribution of the tested statistical property. NIST uses an asymptotic distribution.

The hypothesis is rejected if the p-value is lower than the level of significance of the test (for example 0.01 or 0.001). Thus, a threshold of 0.01 means that one sequence among 100 sequences is expected to be rejected. A p-value greater than this threshold (respectively lower) indicates that the sequence is considered to be random (respectively non-random) with a 99% confidence.

The NIST proposes two methods to decide whether or not a generator is suitable for a cryptographic use. A set of sequences is produced by the generator, and its quality is evaluated by means of statistical tests. The result is determined from the rate of sequences that successfully pass each test (p-value greater than the level of significance), or from the uniformity of the p-values.

Even if tests like the *monobit test*, the *longest runs test*, or the *approximate entropy test* could be theoretically applied on short sequences (100 bits), the recommended length is $20,000$ bits long according to the NIST, because asymptotic approximations are used to determine the limiting distribution.

Moreover, some tests like the *linear complexity test* or the *random excursions test* require at least 10^6 bits to be applied. For short sequences, the NIST suggests that asymptotic distribution would be inappropriate and would need to be replaced by exact distributions that, according to them, would commonly be difficult to compute. Thus, some new tests [23] [8] [2] together with their exact distribution were proposed and a new method to take the decision of randomness for short sequences [24], but they still require a set of sequences to determine the randomness.

3.2 Application of statistical tests in our context

Dumps obtained from embedded devices typically contain information fields whose lengths are between 1 bit and $1,024$ bits (the size of an entire dump

is typically 100-bit to 40,000-bit long). The location and the size of each information is not known, therefore we need to apply statistical tests on overlapping sequences with different sequence lengths. Each sequence tested can be seen as an output of a different generator (name, date, ciphered or hashed data, etc.), then for a dump, only one sequence per generator can be tested. Section 3.1 and the above-mentioned arguments justify that most of statistical tests are not suited to short sequences, and the technique used by the NIST to decide whether or not a sequence is random is therefore not applicable. Moreover, there is no technique that use a combination of statistical tests to take the decision of randomness. In our context, the decision of the classification of each bit into meaningful information or cryptographic material is only done by directly comparing a p-value to a threshold, but this threshold need to be determined.

4 Machine learning for dump analysis

A first step to distinguish meaningful information from cryptographic materials in a memory dump consists in identifying the more appropriate statistical tests and associated parameters. To do so, we use a *boosting* algorithm, namely a machine learning approach that consists in creating a strong classifier by combining weaker ones [13]. The boosting algorithm also determines the best threshold to be used for the classification. In our case, the set of weak learners is a selected set of traditional statistical tests with parameters and a small subset of these tests is our strong learner for the final decision. We have chosen this approach due to its ability to determine the best features to use (i.e. statistical tests in our case). The AdaBoost algorithm [10] is a popular boosting algorithm: it is efficient both in terms of recognition and time computation. This section describes the feature descriptors (parameters of the statistical tests), the definition of data used during the learning and classification steps, then the AdaBoost algorithm and finally a process to merge results of the classification in a set of dumps.

4.1 Feature descriptors

A set of potential parametrised statistical tests $\mathbf{T} = \{T_i,\ i = 1 : N\}$ is described in this section. These tests are our feature descriptors of binary sequences, in which each T_i is defined by T_i = [Test, Sequence length, Shift length, Internal parameters, Selection p-value]. A set of data are parametrised with a label *meaningful information* or cryptographic material $U = \{u_i \in \{1, 2\},\ i = 1 : N\}$ for each bit. All those parameters lead to have a set $\mathbf{T}$ of approximately $2,000$ features. Theses features are defined as follows:

Test: It describes the statistical test that is applied on the binary sequence. All these tests are the NIST tests except those that require 10^6 bits or a specific pattern to test. It represents 8 tests : Monobit, Runs, Block Frequency, Serial, Discrete Fourier Transform, Approximate Entropy, Cumulative Sum, and Longest Runs. These 8 tests provided in the NIST suite are completed by the

Autocorrelation Test [14] and the tests for short sequences: TBT test [2] and Saturation Point Test [23].

Sequence length: It is the length of sequences on which the statistical tests are applied. We have decided to bound those sequences to 32 bits for the minimum length. This value of 32 bits is chosen because tests on too short sequences are not relevant and meaningful information (like date, name, address, etc.) require at least a certain amount of bits to be encoded. Considering that one character is encoded on 5 bits at least (minimum value to encode the 26 latin letters, as used by Mobib cards), sequences of 6 characters are represented with 30 bits. Consequently, sequence lengths used in our experiments are chosen in the set $[33, 41, 49, 56, 69, 77, 85, 96]$. Experiments have shown that considering more than 96 bits is useless.

Shift length: It is the length of the shift between the two start bits of two consecutive tested sequences in a dump. This means that all the bits of the dump are tested $Sequence\ length/Shift\ length$ times, except beginning bits and ending bits. Each bit is tested in several different sequences, and one p-value per test is generated for the bit. For each possible sequence length, 10 shift lengths are used, represented by a percentage of the length : $[10\%, 20\% \dots 90\%]$.

Internal Parameter: For some tests, there are additional inputs. For example, the serial test looks at the proportion of all blocks of m bits in the tested sequence, m is an additional input for the test. So, for each test that requires an additional input, $5 - 10$ values for the parameter are defined. Some values are only used for longer sequences, because when the value grows up, the required sequence length grows up, too.

Selection p-value: Due to the use of a shift, as explained before, each bit has several p-values linked to it. However, to apply the boosting, we need only one p-value per bit. The parameter Selection p-value represents the method that chooses this p-value, it can be the mean, min, max or the geometric mean of the p-values assigned to the bit.

4.2 Data generation for learning and classification

Boosting requires a large number of elements of each class (meaningful information and cryptographic material) from different embedded objects, to be representative of all the existing embedded devices.

As a consequence, although we extracted the data of about 300 devices (using CardPeek [17] or RFIDIOT [16]) from various applications: access control, transportation, credit cards, French health insurance and loyalty cards, train coupons, Belgium e-passports, ski passes, etc. these data cannot be used for the learning phase, because the class of the bit sequences is not known for every card. In order to solve this problem, we decide to set up a large synthetic dump containing data similar to real dumps, inspired by our 300 dumps. This synthetic

dump is modeled by several sequences of variable lengths, separated or not by a repetition of 0 or 1 bits, containing cryptographic materials (hashed or ciphered data, cryptographic keys etc.) or meaningful information (dates, names, etc.).

We have generated a synthetic dump of $40,000$-bit long. It contains approximately 65% of meaningful information, where sequences are between 20 and 80-bit long. Cryptographic materials are generated from various cryptographic algorithms as RSA, AES, SHA-1, etc. They are truncated to obtain the expected length. Meaningful information includes dates with different encoding, e.g., ASCII, BCD, and various formats like YYYY-MM-DD, YY/MM/DD, etc. There are also names, textual information, and postal codes with various encoding techniques.

4.3 AdaBoost

A weak classifier is a statistical test associated with its parameters. The AdaBoost algorithm [10] (adaptive boosting algorithm) aims to identify the best combination of weak classifiers to build a stronger one. A weak classifier can correctly guess whether a sequence is random or not with a probability that is slightly better than 50%. The SAMME.R [27] algorithm (SAMME is for Stagewise Additive Modeling using a Multi-class Exponential loss function and the R for Real) is an improved multi-class variant of the AdaBoost algorithm [21][11] that is used in the next section.

The boosting first selects the best parametrised test to classify the bits of the dump, then misclassified bits receives a more important weight. The booting then selects the second better test taking into account the bit weights, and it then updates the weights. Repeating those actions until all the bits are correctly classified, or a certain number of tests preset have been reached.

4.4 Merging results in a set of dumps

One may expect dumps associated to a given application to be identically structured, i.e., containing the same fields, in the same locations. For example, in Mobib dumps, the name, the birth date and the postal code of the holder or details of its last trips are always similarly located in the dumps. The data of these fields vary for each dump but the class (meaningful information or cryptographic material) is the same. Therefore a classification of the bits of the application is computed rather than a classification for each dump. More precisely, the class of the ith bit of the application is decided by a majority vote on the results of the ith bit of each dump. The merger process is also applied on cyclic records, because they contain the same fields in the same locations. A set of dumps of the same application can be obtained by dumping memory of different cards, or by dumping the cards at different time of the card lifetime.

In reality, the structure of all dumps for a given application is not always identical : some records are possibly missing, or are not of the same length, the number of repetitions of cyclic records can vary or the data stored in the field is not always of the same length and the value of the non-used bits of the

allocated space for the field is uncertain. For example, EMV cards that come from different banks do not store necessary the same number of transactions, do not contain all possible records of the EMV specification, and the field of the name is padded with 0x20 when the name is shorter than the allocated space, etc. Consequently, a pre-processing phase is needed to identify the records in each dump of the application. This operation does not require the knowledge of the card specification, because it is performed by analysing the structure of the data, including the size and the location of the records in the dump, combined with the presence of runs of 0 or 1 separating fields in the record.

5 Experiments

Figure 3 is a summary of the performed experiments to obtain the best statistical tests to distinguish meaningful information and cryptographic materials from dumps. The tested sequence of 6 bits represents a (short) dump. The first step consists in applying each feature T_i (a parametrised statistical test) to all subsequences of the tested sequence, each feature generates a p-value P_i. As subsequences overlap, several p-values are attributed to each bit of the tested sequence, the boosting algorithm requires only one, thus the p-value selection is done, for example the mean p_i of its p-values P_i is attributed to each bit. These two steps are done for each feature and all p-values are stored in an array together with the class of each bit, then the boosting algorithm is applied on it to extract the features that lead to the best classification of the data from the tested dump.

5.1 Learning with AdaBoost

Additionally to the parameters of the statistical tests, there is one parameter to set related to the boosting algorithm such as the number of features used in the classification. The boosting algorithm combines many weak classifiers to obtain the optimal value. It seems unlikely to obtain a classification 100% correct. This parameter limits the number of statistical tests of the set.Regarding their influence on the results, experiments suggest a number of statistical tests between 1 and 15.

Two synthetic dumps are generated, (one represents the learning set and the second the classification one) and the boosting looks for the best set of tests from the learning database. Then, varying the number of tests in the set given by the boosting on the learning dump, each set of tests is applied on the testing dump. The set of tests which leads to the best rate of recognition on the testing base is saved. Experiments have been realised with our own python program using the AdaBoost-SAMME.R algorithm from Scikit-learn [18]. Creation of the p-values array takes several hours to be computed, parallelised on 48 cores processor (on 4 AMD Opteron 6174 2,2 GHz) and uses tens of gigabytes of RAM. Application

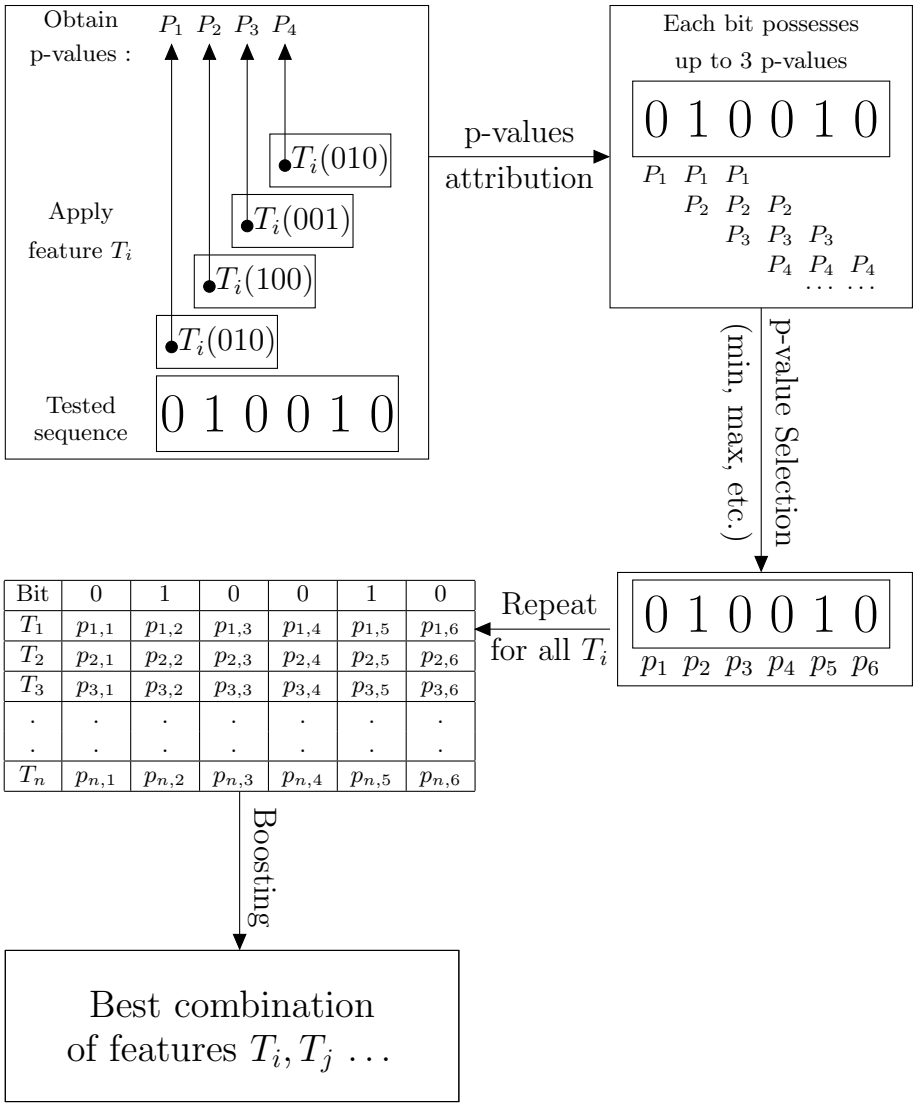

Fig. 3: Boosting example with a test of length 3 bits and a shift of 1 bits on a sequence of 6 bits length

of the AdaBoost algorithm takes between some minutes and two hours depending the number of tests of the set on a single core. The application of the set of statistical tests on a real dump, as described in the next subsection, is almost instantaneous, for big dumps (40,000 bits), it takes few seconds.

The best set of statistical tests obtained after learning with AdaBoost is composed of two tests: approximate entropy and longest runs. The approximate entropy test compares the frequency of overlapping blocks of length m and $m+1$ to the expected result for random data. The longest runs test compares the length of the longest run of ones to the expected result for random data, as detailed in [20]. Parameters of these tests are obtained as following:

- Approximate entropy with a length of 96 bits, a shift of 86 bits between sequences tested, an internal parameters of 2 and the p-value selected is the maximum for each bit.
- Longest Runs with a length of 96 bits, a shift of 86 bits between sequences tested, an internal parameters of 8 and the p-value selected is the mean for each bit.

The recognition rate of these tests is 83.94% (about 87% for cryptographic bits and 82% for bits of meaningful information) on the learning dump and 83.42% (about 80% for cryptographic bits and 85% for bits of meaningful information) on the testing one. The boosting algorithm selects the most pertinent statistical tests in relation to our context of short sequences belonging to memory dumps. Note that, slightly varying the learning data, the boosting Algorithm returns other strong classifier (with different statistical tests and parameters) providing similar recognition rate. It can be surprising to only use two statistical tests but as presented in the following section, those two tests lead to good results on real dumps. Another advantage in the use of only two tests is the efficiency in term of computation time: less than 2 seconds to analyse an EMV dump of 30,000 bits. One can notice that these two statistical tests use 96 bits to take their decision, but they are able to detect the class of sequences that are shorter than 96 bits, because all bits are tested several times due to the shift between tested sequences.

5.2 Recognition on real dumps

In this subsection, the classifier trained on synthetic data is used to classify meaningful information and cryptographic materials on real dumps of memory. This set is applied on more than 30 EMV cards [9], 2 VITALE cards (the french health insurance card) and on 7 Mobib cards. In these cards, the meaning of an important part of the data is publicly known (EMV, VITALE) or a previous work of the authors allows to determine it, so, the ground truth (i.e. theoretical classification of the data) is easily accomplished. It represents more than 600,000 bits of data with 140,000 cryptographic bits and 500,000 bits of meaningful information. As result, we obtain a recognition rate of 86.11% for cryptographic

bits and 96.97% for bits of meaningful information. Figure 4 shows an example of the classification result for an EMV card dump. When the set of tests is applied on Mobib cards, we get between 95% and 100% of recognition rate for each card.

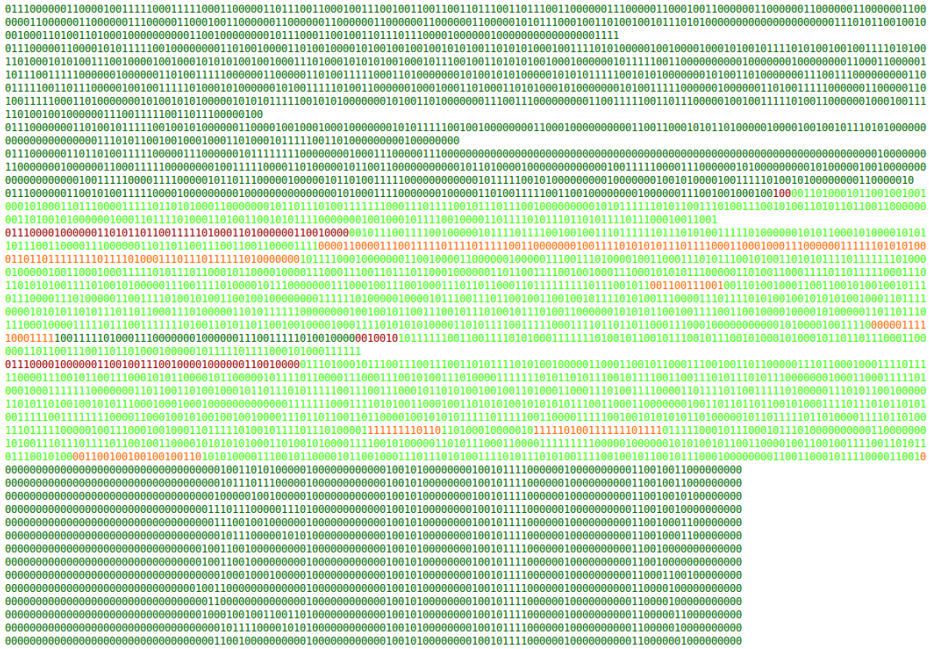

Fig. 4: Example of EMV dump analysis result

- Dark Green: Bits of meaningful information detected as meaningful information
- Light Green: Cryptographic bits detected as cryptographic
- Orange: Cryptographic bits detected as meaningful information
- Red: Bits of meaningful information detected as cryptographic

Some further analysis can improve the results, for example if a single cryptographic (or from meaningful information) bit is surrounded by a significant amount of meaningful information (or cryptographic) bits, then this bit is certainly misclassified. Errors are often localised on the transition between a cryptographic sequence and a meaningful one. In Mobib cards, only one field is not always recognised as meaningful information (the name of the holder of the card where each letter is encoded with 5 bits). It should be noted that in Mobib card, there is only meaningful information, and the sequences are quite short, most of them are shorter than 100 bits. Table 1 presents the obtained results on real

dumps with the proposed approach.

Table 1: Detection of cryptographic bits and bits of meaningful information in EMV, VITALE and Mobib dumps

Dump type	Cryptographic bits	Detected bits	Meaningful bits	Detected bits
EMV	131,384	85.97 %	379,352	95.95 %
VITALE	9,168	88.14 %	126,160	99.93 %
Mobib	0	–	9,681	98.34 %

5.3 Merging results on EMV cards

As seen in the previous section, the classification of meaningful information and cryptographic bits by the proposed approach provides good results. While the experiments for Mobib dumps exhibit a successful rate of 100%, the EMV ones gives around 90%. Therefore, we propose to use the merger process to improve the last results.

Since all our EMV dumps do not contain same records, therefore we have to firstly locate and analyse the structure and the presence of some fields into our EMV dumps, which is not a trivial task. These fields are information about the holder, the card or cryptographic materials. Fields having already a recognition rate equal to 100% are discarded for this merger experiment. In the following experiments, 10 fields representing in total 3,560 bits are selected, split as 3,312 cryptographic bits and 248 bits of meaningful information. Since these fields are repeated numerous time in each dump and our database is composed of 34 EMV dumps, our merger process has to classify 21,120 bits of meaningful information and 124,512 cryptographic bits.

Applying the merger process for all these fields on 34 EMV dumps, we obtain nearly 100% of rate of recognition. More precisely, only two bits of a cryptographic field (a certificate of 1,024 bits) are not recognised. We experimented a second time this process by merging the minimum of results in order to obtain a score nearly of 100%. More precisely, for fields that are between 16 and 256-bit length, with a mean of rate of recognition on each dump between 80% and 95%, only three scores are needed to be merged to obtain a score of 100% regardless the class. For fields that are longer, about 1,000 bits with a mean of recognition rate on each dump of 85%, more scores are needed to obtain a score of 100%. Except for one field, with only 3 merged results, we improve the recognition by 10 points, that is reaching 95%, and with 10 merged results, we obtain the score of 100%. For the fields that we never obtain 100%, we improve the recognition rate up to 97% with only 10 merged results. Due to the use of a majority vote,

Table 2: Fusion results by field

Field name	Class	Length (bits)	Mean result	Fusion	Sequences merged
Issuer PK Certificate	2	1,024	85.85 %	99.8 %	22
Signed Static App. Data	2	960	87.04 %	100 %	5
ICC PK Certificate	2	1,024	84.21 %	100 %	9
ICC PK Remainder	2	144	82.32 %	100 %	3
Issuer PK Remainder	2	160	87.16 %	100 %	3
Cardholder Name	1	168	97.54 %	100 %	3
App. Label	1	16	89.0 %	100 %	3
App. Preferred Name	1	16	83.23 %	100 %	3
App. Effective Date	1	24	97.6 %	100 %	3
App. Expiration Date	1	24	99.53 %	100 %	3

it is better to merge an odd number of results, above all if we merge few results.

Table 2 presents the fusion result for each selected field from EMV cards, where the class is 1 for meaningful information and 2 for cryptographic material, the mean result is the mean of recognition rates of the analysis applied on each dump separately. The Fusion represents the best recognition rate after applying the Fusion, and sequences merged is the required number of sequences merged to obtain the best result.

6 Conclusion and perspectives

This paper investigates memory carving techniques for embedded devices. Given that cryptographic material in memory dumps makes carving techniques inefficient, we introduce a methodology to distinguish meaningful information from cryptographic material in memory dumps. Our approach is based on an adaptive boosting algorithm based on statistical tests for randomness. We propose to combine several weak classifiers to build a stronger one. Experiments return a detection of meaningful bits from 95.95% of success for EMV cards to 99.93% for VITALE ones, and a cryptographic bits recognition greater than 85%. We also introduced a fusion strategy to improve the previous results, merging the classification of several dumps of the same application. With only 3 merged dumps, the experiments exhibit a recognition rate of 99.8%.

Future work includes an automatic interpretation of meaningful information of the dumps, by retrieving names, dates, address, etc. Another possible work is the classification of dumps into several categories to adjust the interpretation method.

References

[1] AG, S.: Skidata, `http://www.skidata.com/en.html`

[2] Alcover, P.M., Guillamón, A., Ruiz, M.d.C.: A new randomness test for bit sequences. Informatica 24(3), 339–356 (2013)
[3] Avoine, G., Kalach, K., Quisquater, J.J.: epassport: Securing international contacts with contactless chips. In: Financial Cryptography and Data Security, pp. 141–155. Springer (2008)
[4] Burdach, M.: Physical memory forensics (2006), https://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Burdach.pdf
[5] CNA, C.: Calypso, http://www.calypsostandard.net/
[6] Cohen, M.: Advanced carving techniques. Digital investigation 4, 119–128 (2007)
[7] Coisel, I., Sanchez, I., Shaw, D.: Physical attacks against the lack of perfect forward secrecy in dect encrypted communications and possible countermeasures. In: International Wireless Communications and Mobile Computing Conference (IWCMC). pp. 594–599 (2015)
[8] Doğanaksoy, A., Çalık, C., Sulak, F., Turan, M.S.: New randomness tests using random walk. In: National Cryptology Symposium II (2006)
[9] EMVCo: EMV integrated circuit card specifications for payment systems (Jun 2008), application Independent ICC to Terminal Interface Requirements
[10] Freund, Y., Schapire, R., Abe, N.: A short introduction to boosting. Journal-Japanese Society For Artificial Intelligence 14(771-780), 1612 (1999)
[11] Friedman, J., Hastie, T., Tibshirani, R., et al.: Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). The annals of statistics 28(2), 337–407 (2000)
[12] Friedman, W.F.: The index of coincidence and its applications in cryptanalysis. Aegean Park Press (1987)
[13] Kajdanowicz, T., Kazienko, P.: Boosting-based sequential output prediction. New Generation Computing 29(3), 293–307 (2011)
[14] Knuth, D.E.: The art of computer programming, vol. 2: Seminumerical algorithms, revised edition (1969)
[15] Lanet, J.L., Bouffard, G., Lamrani, R., Chakra, R., Mestiri, A., Monsif, M., Fandi, A.: Memory forensics of a java card dump. In: Smart Card Research and Advanced Applications, pp. 3–17. Springer (2014)
[16] Laurie, A.: Rfidiot, http://rfidiot.org/
[17] Pannetrat, A.: Cardpeek, http://pannetrat.com/Cardpeek/
[18] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. The Journal of Machine Learning Research 12, 2825–2830 (2011)
[19] Poisel, R., Tjoa, S.: A comprehensive literature review of file carving. In: Availability, Reliability and Security (ARES), 2013 Eighth International Conference on. pp. 475–484. IEEE (2013)
[20] Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., Vo, S.: A statistical test suite for random and pseudorandom number generators for cryptographic applications. Tech. rep., DTIC Document (Apr 2010)
[21] Schapire, R.E.: The boosting approach to machine learning: An overview. In: Nonlinear estimation and classification, pp. 149–171. Springer (2003)
[22] Shamir, A., Van Someren, N.: Playing hide and seekwith stored keys. In: Financial cryptography. pp. 118–124. Springer (1999)
[23] Sulak, F.: A new statistical randomness test: Saturation point test. International Journal of Information Security Science 2(3), 81–85 (2013)

[24] Sulak, F., Doğanaksoy, A., Ege, B., Koçak, O.: Evaluation of randomness test results for short sequences. In: Sequences and Their Applications–SETA 2010, pp. 309–319. Springer (2010)
[25] Van Deursen, T., Mauw, S., Radomirovic, S.: mCarve: Carving attributed dump sets. In: USENIX Security Symposium (2011)
[26] Yoo, B., Park, J., Lim, S., Bang, J., Lee, S.: A study on multimedia file carving method. Multimedia Tools and Applications 61(1), 243–261 (2012)
[27] Zhu, J., Rosset, S., Zou, H., Hastie, T.: Multi-class adaboost. Ann Arbor 1001, 48109 (2006)