Chapter 1

Distinguishing encrypted from non-encrypted data

Eric Järpe and Quentin Gouchet

Abstract

Discriminating between encrypted and non-encrypted information is desired for many purposes. Much of the efforts in this direction in the literature is focused on deploying machine learning methods for the discrimination in streamed data which is transmitted in packets in communication networks. Here, however, the focus and the methods are different. The retrieval of data from computer hard drives that have been seized from police busts against suspected criminals is sometimes not straightforward. Typically the incriminating code, which may be important evidence in subsequent trials, is encrypted and quick deleted. The cryptanalysis of what can be recovered from such hard drives is then subject to time-consuming brute forcing and password guessing. To this end methods for accurate classification of what is encrypted code and what is not is of the essence. Here a procedure for discriminating encrypted code from non-encrypted is derived. Two methods to detect where encrypted data is located in a hard disk drive are detailed using passive change-point detection. Measures of performance of such methods are discussed and a new property for evaluation is suggested. The methods are then evaluated and discussed according to the performance measures.

Keywords: Likelihood ratio, change-point detection, cryptology, compression, uniform distribution

1. Introduction

The discrimination of encrypted data from other kinds of data is of interest in many areas of application. For instance for making other applications work for the communication traffic in a network where the means for application may depend on whether the traffic data is plaintext/cleartext, compressed, encrypted or encoded in some way. Also, there may be security reasons, e.g. the uncontrolled flow of encrypted data of which some may be transmitted for malicious purposes could be argued in need of supervision network abilities. To these ends, various methods, mainly of machine learning, have been suggested through the last decades.

1.1 Methods for discrimination in case of streamed data

As a foundation for the treatment, some kind of preprocessing is due. Among methods for this step are the chi-square statistic, Shannon entropy,

Kolmogorov-Smirnov statistic, Lilliefors test, Cramér-von Mises statistic, discrete runs test, autocorrelation's test and the index of coincidence test to mention some [15, 19].

In [15] the problem of filtering encrypted data upon traffic monitoring which is outbound from a computer or a network, so-called egress filtering, is considered. Many aspects and properties of the problem are brought to attention and an extensive account for various evaluation techniques is mentioned.

Different measures for discriminating a given distribution from the normal distribution are the topic in [19]. For discrimination between encrypted and non-encrypted data other distributions than the normal are more relevant but some of the measures considered in this paper can also be used for the more general case.

In [10] machine learning methods are deployed to a fully automated method to the distinction between encrypted and compressed data, claimed to be the first of its kind. A dataset for further evaluation and benchmarking is also provided.

A method for discriminating between compressed and encrypted data in the case when the data is voice over IP with constant and variable bit rate codecs is proposed in [6]. The method is evaluated utilizing the NIST [3] and the ENT [24] test suits.

[13] does not suggest a solution to the discrimination problem but rather a method for fast and distribution true simulation of data reflecting the properties of encrypted and compressed data respectively.

In [25] a classification procedure, based on Gaussian mixtures and hidden Markov models, is detailed. An elaborate comparative evaluation is carried out taking 9 other attempts to solve the same problem into account. The study concludes that the proposed method renders a better classification at a lower computation complexity.

A lucid convolution neural network method to classify data into being encrypted or non-encrypted is presented by [14]. The proposed method is thoroughly evaluated for various kinds of network traffic and some different performance measures in the field of machine learning.

1.2 Methods for discrimination in case of static data

A different need for discrimination between encrypted and unencrypted data is the following. In police investigations concerning heavy criminality and organized crime where evidence of criminal activity is residing as data on a computer hard disk drive (HDD) the capability to distinguish encrypted files from non-encrypted ones may be primordial [1, 21, 22]. When the police seize an HDD containing data belonging to a suspected criminal, that data can be material of evidence in a subsequent trial. But criminals often try to make sure that the police will be able to use that data. Possible actions to obstruct data access are then to encrypt and/or to *quick delete* that data. In case of quick delete of data, the pointers to the files are destroyed but the content of the files is still left. The other action is to encrypt files. Using state of the art tools for encryption of files (such as VeraCrypt, Bitlocker, Ciphershed and thereby provided cyphers) there is no general procedure of breaking the encryption other than brute force attack by guessing the cypher algorithm and systematically guessing the encryption key [4]. If some of the files are encrypted and others not, it is then a delicate matter to distinguish between the two if the code has been quick deleted. Nevertheless, the importance of

discriminating between encrypted code and plaintext is also stressed by the fact that brute force attacking all clusters of data on the hard drive would be an impossible task while being able to separate the encrypted code from the non-encrypted would make a substantial improvement of the chances for successful cryptanalysis. The police authorities usually have software to brute force those encrypted data but this procedure may be very time consuming if the amount of data is so large that different parts of it have been encrypted with different keys. In juridical cases, time is typically of the essence since the chances of success in prosecution and proceedings against a criminal is depending on deadlines (such as time of arrest, time of trial etc) any time savings in the procedure of extracting evidence from the HDDs is essential. Thus time has to be spent on the appropriate tasks: code-breaking only on the encrypted data rather than trying to decypher non-encrypted data. Given a single file, the task of determining whether it is encrypted or not is usually easy; but given a whole HDD without knowing where the encrypted files are, this is trickier.

There are several software solutions for indicating whether a file is encrypted or not, mostly checking the header of the file and looking for some known pattern like the EFS (Encryption Files System on Windows), BestCrypt, or other softwares' headers. Such alternatives cannot be used in the case when the user has performed a (quick) delete of the HDD because then the pointers to all files are lost. Nevertheless, the files remain on the HDD: upon deleting a file on an HDD, the pointers to the beginning and end of the physical space on the HDD containing the information of the file are removed. Still, the physical space on the HDD containing the information of the file is not overwritten because that operation would be slow, i.e. as slow as writing the same file again. The operating system's designers rather leave the file in the HDD intact but indicate that this location is free to host some other data. If nothing has been overwritten, this means that the file is still stored in the HDD for some time which allows recovery software to recover those files.

Recovery software might help to simply recover the files but might also overwrite the data contained in HDD which could result in loss of evidence in case of an investigation. In this case, the police would have to locate the encrypted files without using recovery software or "header-detector" software.

Here methods to locate quick deleted encrypted data are presented and detailed. First, a description of how encrypted data is different from other data is presented. This is followed by the introduction to statistical change-point methods for discrimination between encrypted and non-encrypted data. Finally, the results of these procedures are presented along with some experimental values to evaluate the methods.

However, such a method will only work on mechanical HDDs and not with flash memory devices: in flash memory (like USB memory sticks or Solid State Drive (SSD)), as soon as a file is removed it is erased from the memory because data cannot be overwritten. Therefore as soon as data is deleted, the operating system will choose to delete the pointers and the data to gain time for the time when the user will decide to save data at this location. But erasing in flash memory also takes longer since the pointer have to be deleted and all the files removed which is as long as copying new files on the device.

This application differs from the one mentioned in Subsection 1.1. In the previous case when data is commonly considered being transmitted in packets in a network while here, there is static access to the data. In the previous situation, data consists of sometimes small files where statistical methods for making a foundation indicating if the data is encrypted or not becomes

weaker [17] as opposed to the situation here where there is a large amount of data possibly of both kinds, encrypted and non-encrypted. In the previous case with dynamic data in all senses – variable in size, access, time, kind – methods of machine learning which could pick up on the current circumstances were preferable while here, in the case of fixed data, fast and efficient methods of change-point detection becomes a far more advantageous choice. The machine learning methods need training data while the change-point methods can start from scratch with pre-defined parameter values optimized for the situation or with very little run-in data for calibrating the levels. The performance of the machine learning methods is still a matter of research since these are very highly structured procedures, sometimes black box techniques impossible to fully evaluate all properties of, while the change-point detection methods are long since optimized and very thoroughly evaluated from all kinds of aspects of performance through a much longer time.

2. Method

2.1 Description

If encrypted data is not uniformly distributed, the cryptosystem used to cypher those data has a bias and can therefore be attacked. For this reason, characters of the cyphertext produced by any modern high-quality cryptosystem are uniformly distributed [2], [23] i.e. the values of the bytes of the cyphertext are uniformly distributed on some character interval. Indeed, a cryptosystem that does not have this property would be weak since it would be possible to attack it on this bias (distinguishing attacks such as on the RC4 encryption algorithm). The other types of files do not possess this feature although the contents of some types of files are close to being uniformly distributed. The files coming closest without being encrypted are compressed/zipped files: those files are indeed very close to cypher files in terms of the distribution of their character's byte numbers. Albeit small there is a difference in distribution making it possible to tell compressed files and encrypted files apart.

A technique to quantify this distribution difference is by using chi-square statistic and more or less performing a chi-square test (see [18]) to tell whether the data is suspiciously uniform or not. Another classical method is to calculate the Kolmogorov-Smirnov distance, see e.g. [19**?**] for a more extensive account of ways to indicate whether data are encrypted or not. Procedures building on such preprocessors can then be defined.

One attempt to exploit this distribution difference utilizing the chi-square statistic is [5] where a method to automate the discrimination between encrypted and non-encrypted (i.e. most critically compressed) data into a method with impressing performance. Another approach could be to use means of anomaly detection in the theory of machine learning to develop an adaptive method. The proposed method, however, stems from using statistical change-point detection. The anticipated advantage with this could be the mathematically proven optimality with these methods in terms of efficiency and accuracy under the given assumptions. For many applications, the assumptions of entirely relying on the distribution of the data, in-control and out-of-control are commonly a subject of great controversy. Here, the situation is different though, since the hard drive and its data is not a dynamic system

where the content changes its distribution owing to outer time-depending circumstances.

Before going into detail about these methods the preprocessing techniques are introduced.

2.2 Distribution of encrypted data

The working hypothesis is that data (i.e. characters) constituting encrypted files are uniformly distributed, while data of non-encrypted files are not (i.e. differently distributed depending on which type of non-encrypted files). The goal now is to be able to tell an encrypted file from a non-encrypted one.

Let us assume the data constitutes of characters divided into clusters, $c_1, c_2, c_3, \ldots$ with N characters in each cluster. The characters that are used range over some alphabet of a set of possible forms. Merging these forms into K classes, the counts O_{kt} of occurrences of class k characters in cluster t are observed. One method of measuring distribution agreement is by means of a χ^2 test statistic, $Q_t = \sum_{k=1}^{K}(O_{kt} - E_{kt})^2/E_{kt}$ where E_{kt} are the expected counts of occurrences of characters in class k, cluster t. Under the hypothesis of uniformly distributed characters, the expected counts of occurrences within each class is $E_{kt} = \frac{N}{K}$ reducing the statistic simplifies to

$$Q_t = \frac{K}{N}\sum_{k=1}^{K} O_{kt} - N$$

the values of which are henceforth referred to as Q scores. Large Q scores indicate deviance from the corresponding expected frequencies E_{kt}. The smallest possible Q score being 0 would be attained if all $E_{kt} = O_{kt}$. The expected value, E_{kt}, in each class, should not be smaller than 5 for the statistic to be relevant (5 is a commonly used value; other values like 2 or 10 are sometimes used depending on how stable the test statistic should be to small deviances in tail probabilities). Therefore one should use at least 5 kB of data in each cluster to enable this test to be relevant. But the larger the number of bytes that are in each cluster, the worse the precision to detect encrypted file values: indeed, if too many bytes were detected as being unencrypted (but were actually encrypted) a large amount of encrypted data might not be detected in the procedure.

Here, the alphabet used was the numbers $0, 1, \ldots, 255$ representing the possible size in bytes of the characters in the data. These numbers were divided into $K = 8$ classes (class 1: values of bytes in $[0, 31]$ to class 8: values of bytes in $[224, 255]$) and the clusters of size $N = 64$ bytes making the expected values in each class $E_{kt} = 8$. Assuming that encrypted data are uniformly distributed, the Q scores based on counts of characters in encrypted data are χ^2 distributed, see Figure 1 and since 8 classes were chosen, the number of degrees of freedom is $8 - 1 = 7$.

2.3 Distribution of non-encrypted data

For non-encrypted data, the distribution is more complicated. Each type of file has its distribution. Consequently, the standardized squared deviances from expected counts under the assumption about uniform distribution are larger and so are the Q scores of the χ^2 statistic. However, two problems emerge. Firstly, the size of these increased deviances depends on the type of data – i.e.

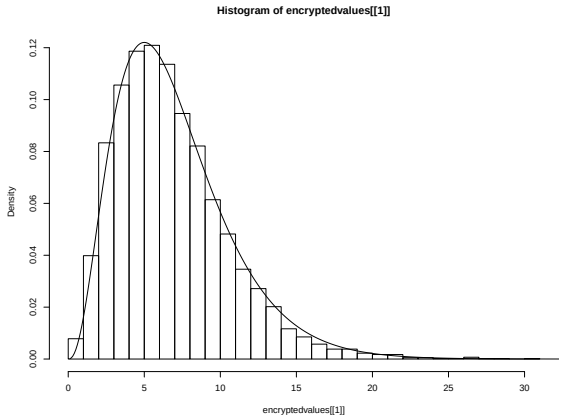

Figure 1.
Distribution of the Q scores of encrypted files (obtained by using more than 5000 files) with the distribution function of $Q \in \chi^2(7)$

is the data a text file, an image, a compiled program, a compressed file or some other kind of file? – and how should this information be properly taken into account? Secondly, what is the distribution of the Q score in the case when the data are not encrypted?

To develop a method for distinguishing between encrypted and non-encrypted data it is sufficient to focus on the non-encrypted that is most similar to the encrypted and this turns out to be compressed data. Other types of files such as images, compiled programs etc. commonly render higher Q scores and are therefore indirectly distinguished from encrypted data by a method that is calibrated for discriminating between encrypted and compressed data. About the second question, this is not readily answered. Rather, we just suggest modelling the Q score as being scaled χ^2 distributed, i.e. the Q score is assumed to have the distribution of the random variable αX where $\alpha > 1$ and $X \in \chi^2$. The validity of this approach is sustained by empirical evaluation based on more than 5000 compressed files. The resulting empirical distribution of their Q scores and the distribution of αX where X is χ^2 distributed and the value of $\alpha = 1.7374$ was estimated by the least square method was plotted in Figure 2.

2.4 Change-point detection

Change-point detection [7–9, 11, 12] is a field of mathematical statistics where the object is to quickly and accurately detect a shift in distribution from on-line observation of a random process. This can be done actively (stop collecting the data as soon as a shift is detected) or passively (continue collecting the data even if a shift is detected in order to detect more shifts). Here passive on-line change-point detection was used to detect if the data from an HDD shifts from non-encrypted to encrypted and vice versa. The change-point detection method is a stopping rule

$$\tau = \inf\{t > 0 : a_t > C\}$$

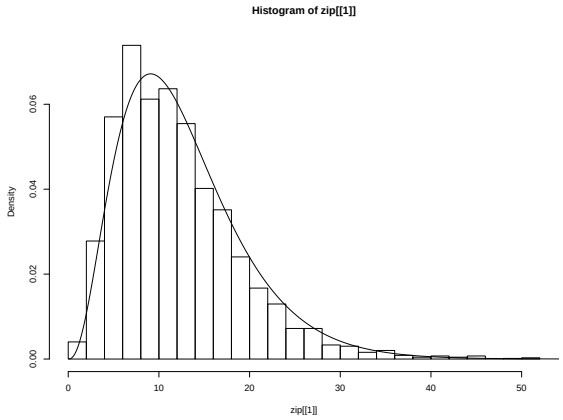

Figure 2.
Distribution of the Q scores of compressed files (obtained by using more than 5000 files) with the distribution function of $Q = 1.7374 \cdot X$ *where* $X \in \chi^2(7)$

where a_t is called alarm function and C threshold. The design of the alarm function defines different change-point detection methods while the values of the threshold reflects the degree of sensitivity of the stopping rule. The alarm function may be based on the likelihood ratio

$$L(s,t) = \frac{f_{Q_t}(\boldsymbol{q}_t \mid \theta = s \leq t)}{f_{Q_t}(\boldsymbol{q}_t \mid \theta > t)}$$

where $f_{Q_t}(\boldsymbol{q}_t|A)$ is the conditional joint density function of the random variables $(Q_1, \ldots, Q_t) = \boldsymbol{Q}_t$ given A and where $\boldsymbol{q}_t$ is the vector of the observed values of $\boldsymbol{Q}_t$. Assuming indpendence of the variables $Q_1, \ldots, Q_t$ the likelihood ratio simplifies to

$$L(s,t) = \prod_{u=s}^{t} \frac{f_1(q_u)}{f_0(q_u)}$$

where $f_0(q_u)$ is the marginal conditional density function of Q_u given that the shift has not occurred by time u and $f_1(q_u)$ is the marginal conditional density function of Q_u given that the shift occurred in or before time u.

The conditional density function of the Q score at time t given that the data is encrypted (i.e. uniformly distributed) is

$$f_{\mathrm{E}}(q_t) = \frac{q_t^{k/2-1} e^{-q_t/2}}{2^{k/2}\Gamma(k/2)}$$

where k is the number of degrees of freedom, i.e. the number of classes (which in this study is 8 as explained above).

For the non-encrypted files, the conditional density function of the Q score is modeled by αX where $X \in \chi^2(k)$ and $\alpha > 1$ supposedly reflecting the inflated

deviances from the uniform distribution had the data been encrypted. Thus

$$\begin{aligned} f_{\mathrm{NE}}(q_t) &= \tfrac{\partial}{\partial q_t}\mathbb{P}(\alpha X < q_t \mid X \in \chi^2(k)) \\ &= \frac{(\frac{q_t}{\alpha})^{k/2-1}e^{-q_t/2\alpha}}{\alpha 2^{k/2}\Gamma(k/2)} \end{aligned}$$

is the density function of non-encrypted data Q score. This means that two cases of shift in didstribution are possible:

- Shift from non-encrypted to encrypted data in which case

$$L(s,t) = \prod_{u=s}^{t} \tfrac{f_{\mathrm{E}}(q_u)}{f_{\mathrm{NE}}(q_u)} = \alpha^{k(t-s+1)/2} \exp\Big(- \tfrac{\alpha-1}{2\alpha} \sum_{u=s}^{t} q_u \Big)$$

- Shift from encrypted to non-encrypted data in which case

$$L(s,t) = \prod_{u=s}^{t} \tfrac{f_{\mathrm{NE}}(q_u)}{f_{\mathrm{E}}(q_u)} = \alpha^{-k(t-s+1)/2} \exp\Big(\tfrac{\alpha-1}{2\alpha} \sum_{u=s}^{t} q_u \Big)$$

To detect whether the shift in distribution has occurred or not according to the stopping rule τ mentioned above, an alarm function should be specified. Two of the most common choices here are:

- CUSUM [16]: $a_t = \max_{1\le s\le t} L(s,t)$,
- Shiryaev [20]: $a_t = \sum_{s=1}^{t} L(s,t)$.

Other possible choices are e.g. the Shewhart method, the Exponentially Weighted Moving Average (EWMA), the full Likelihood Ratio method (LR) and others, see e.g. [8] for a more extensive presentation of different methods.

For the CUSUM alarm function, as $\arg\max_{1\le s\le t} L(s,t) = \arg\max_{1\le s\le t} \ln L(s,t)$ the alarm function is simplified without any loss of generality by using the log likelihood values instead.

For both cases the alarm functions can be expressed recursively which facilitates collecting and treating the data as follows.

- The alarm function for shift from non-encrypted to encrypted data for the
  - CUSUM method is

$$a_t = \begin{cases} 0 & \text{for } t = 0 \\ \max\left(a_{t-1} + \frac{k\ln\alpha}{2}, 0\right) + \frac{1-\alpha}{2\alpha} q_t & \text{for } t = 1,2,3,\ldots \end{cases}$$

  - Shiryaev method is

$$a_t = \begin{cases} 0 & \text{for } t = 0 \\ \alpha^{k/2} e^{\frac{1-\alpha}{2\alpha} q_t}(1 + a_{t-1}) & \text{for } t = 1,2,3,\ldots \end{cases}$$

- The alarm function for shift from encrypted to non-encrypted data for the

Results

ν	Methods							
	CUSUM				Shiryaev			
	100	500	2 500	10 000	100	500	2 500	10 000
0.2	4.7844	7.1087	9.5520	11.6905	4.9672	7.3116	9.7634	11.9159
0.15	4.7674	7.0788	9.5109	11.6495	4.8933	7.2409	9.6760	11.8015
0.07	4.7278	7.0162	9.4401	11.5695	4.7455	7.0610	9.4786	11.6114
0.05	4.7176	7.0017	9.4224	11.5420	4.7021	6.9975	9.4308	11.5441
0.02	4.6957	6.9712	9.3860	11.4940	4.6422	6.9144	9.3175	11.4477
0.01	4.6870	6.9581	9.3698	11.4693	4.6150	6.8633	9.2743	11.3973

Table 1.
Values of expected delays ED for the CUSUM and Shiryaev methods for values of $ARL^0 = 100, 500, 2\,500, 10\,000$ *for a shift from encrypted to compressed data*

– CUSUM method is

$$a_t = \begin{cases} 0 & \text{for } t = 0 \\ \max\left(a_{t-1} - \frac{k\ln(\alpha)}{2}, 0\right) + \frac{\alpha-1}{2\alpha} q_t & \text{for } t = 1, 2, 3, \ldots \end{cases}$$

– Shiryaev method is

$$a_t = \begin{cases} 0 & \text{for } t = 0 \\ \alpha^{-k/2} e^{\frac{\alpha-1}{2\alpha} q_t} (1 + a_{t-1}) & \text{for } t = 1, 2, 3, \ldots \end{cases}$$

3. Results

3.1 Evaluation

To quantify the quality of different methods, the performance is compared regarding relevant properties such as the time until a false alarm, delay of a motivated alarm, the credibility of an alarm and so on. The threshold is commonly calibrated against the Average Run Length ARL^0 which is defined as the expected time until an alarm when no parameter shift occurred (which means that this is a false alarm). It is crucial to have the right threshold values for the methods to perform as specified. Setting the threshold such that ARL^0 is $100, 500, 2\,500$ and $10\,000$ respectively (the most common values here are $ARL^0 = 100$ and 500 but the higher values are also considered since the value of ARL^0 defines the number of clusters/time points that are treated before a false alarm and the shift could occur very far into the HDD) properties of the methods regarding delay and credibility of a motivated alarm can be compared. Of course, if the threshold is low, this will lead to more false alarms (detection of a change when there is none) but setting a too high threshold will lead to a drop of sensitivity of the method to detect a shift (higher delay between a shift and its detection) and consequently an increased probability of missing a real shift in distribution.

Usually the expected delay, $ED(\nu) = E_\nu(\tau - \theta \mid \theta < \tau)$ (expectation of the delay of a motivated alarm; see Figure 1) or Conditional Expected Delay $CED(t) = E(\tau - \theta \mid \tau > \theta = t)$ (expectation of the delay when the change point is fixed equal to t) are very important because, for example in health care, the goal is to quickly detect problems to be able to save lifes.

However, in the case of detecting encrypted code, expected delays are less relevant as a measure of performance since the data can be handled without any time aspect: the goal is to detect accurately where the encrypted data is located. A method with high expected or conditional expected delay merely means a slightly less efficient procedure.

A more relevant performance indicator, in this case, is for instance the predictive value $PV = P(\theta < \tau)$ (the probability that the method signal alarm when the change-point has occurred; see Figures 5 and 5) or the percentage of detected encrypted files that is discovered while running the process and how to improve it (see Figure 2).

While running the process, the method will stop at some time, τ, and then estimate the change point, θ, by maximizing the likelihood function by using the data after the last previous alarm and the newly detected change-point. This estimated change-point, $\hat{\theta}$, can be either before or after the true change-point θ. One could increase the intervals where encrypted data were discovered. This would lead to missing less encrypted data (see Figure 2) but also brute-forcing more non-encrypted data.

i	Percentage	
	CUSUM	Shiryaev
0	0.960254	0.961280
1	0.971053	0.971274
2	0.976242	0.978665
3	0.979266	0.980872
4	0.983681	0.985597
5	0.986248	0.986101
10	0.990101	0.990101
50	0.993371	0.994931
100	0.994326	0.995682

Table 2.
Percentage of encrypted files that are detected when the interval of detected change points $[\tau_1, \tau_2]$ is arbitrary replaced by $[\tau_1 - i, \tau_2 + i]$. Typically with large i, the values are very close but not exactly equal to 1: this happens because the change points are very close (i.e less than 10 clusters for example) and the method does not detect any change. Then no cyphertext is detected at all.

Therefore the difference between the change-points and the alarms according to the method is calculated. Since the proportion of encrypted data relative to the total amount of data on the HDD is unknown, the expected proportion of error is suggested. This is to say, given two change-points, θ_1 and θ_2, and two stopping times, τ_1 and τ_2, the expected proportion of error is $E\left(\frac{|\tau_1-\theta_1|+|\tau_2-\theta_2|}{\theta_2-\theta_1}\right)$. However, this value has a sense only when there are no false alarms between τ_1 and τ_2.

If there are false alarms between τ_1 and τ_2, the proportion of undetected encrypted data was added to the proportion of error to determine the proportion of the error made relative to the size of the encrypted data. Assuming that there are n false alarms $\tau_1' < \ldots < \tau_n'$ in $[\tau_1, \tau_2]$, called *expected inaccuracy*, or *EI* for short, is defined as follows.

$$EI = E\left(\frac{|\tau_1 - \theta_1| + |\tau_2 - \theta_2|}{\theta_2 - \theta_1} + \sum_{i=1}^{n/2} \frac{\tau_{2i}' - \tau_{2i-1}'}{\theta_2 - \theta_1}\right) \quad (1)$$

Results

ν	Percentage	
	CUSUM	Shiryaev
0.20	0.113791	0.116934
0.15	0.112024	0.112757
0.07	0.095589	0.096884
0.05	0.086831	0.088897
0.02	0.062135	0.062066
0.01	0.044261	0.043975
0.005	0.030960	0.030376
0.001	0.016548	0.016028

Table 3.
Expected inaccuracy, EI (see Equation 1 below), for different values of the parameter ν and for the two methods. For the application of discriminating between encrypted and non-encrypted data on a harddrive this may be considered to be reflecting the degree of fragmentation of the disk; the less fragmentation the farther apart are the change-points and thus the smaller the ν value, and vice versa.

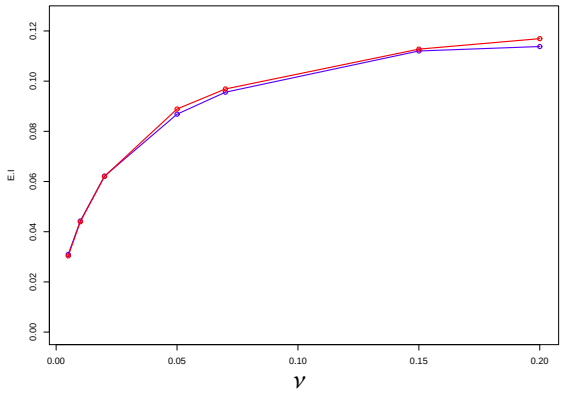

Figure 3.
Expected inaccuracy EI for the CUSUM (blue graph) and Shiryaev (red graph) procedure. One can see that the Shiryaev procedure is little less accurate when ν increases but slightly more accurate for small ν than the CUSUM procedure.

The *EI* was measured for different values of the parameter ν in the geometrical distribution of the change-points for different methods (see Table 3 and Figure 3).

3.2 Complete procedure

The complete process is the method returning a segmentation separating suspiciously encrypted data and most likely non-encrypted data of an HDD; information to be further used to target the brute force cryptanalysis efficiently. This procedure runs a likelihood ratio based change-point detection method and as soon as it detects a change, calculates the maximum likelihood estimator of the change-point to determine where the change-point most likely is located. It will then start over from the location of this estimated change-point with the same method for online change-point detection except that the likelihood ratio is reversed modifying the alarm function to fit the opposite change-point situation, and so on.

ARL^0	Thresholds			
	CUSUM		Shiryaev	
	NE → E	E → NE	NE → E	E → NE
100	1.2260	4.5801	64.0313	44.1271
500	2.6529	6.1250	323.0625	221.8125
2 500	4.2188	7.7120	1618.219	735.4088
10 000	5.5296	9.0990	6475.0547	4441.8413

Table 4.
Values of the thresholds for the CUSUM and Shiryaev methods for $ARL^0 = 100, 500, 2\,500, 10\,000$ specified for detecting a shift from non-encrypted to encrypted data (indicated by NE → E for short) and for shift from encrypted to non-encrypted data (indicated by E → NE) respectively.

3.3 Thresholds and experimental values

The first step is to determine the thresholds rendering $ARL^0 = 100, 500, 2\,500$ and 10 000, for both the CUSUM and Shiryaev methods, for a shift from encrypted to non-encrypted and vice versa. The change-points are commonly geometrically distributed with parameter ν. Here the average time before a change-point is expected to be rather high (several hundred or thousand maybe) as the method deal with 64-byte clusters in an HDD of surely several hundreds of Giga or Tera Bytes. Thus, since $E(\tau) = 1/\nu$, the focus is on very small values of ν for the methods to be reasonably sensitive.

Commonly values of ARL^0 are 100 or 500 to make other properties relevant for comparisons. In the case of this application, however, large values of ARL^0 as 2 500 and 10 000 are studied because the first change-point might not occur until far into the HDD. Adjusting the threshold by simulating data can take very long if ARL^0 is large (2 500 or 10 000, especially for the Shiryaev method). In this case, it can take several hours or even up to days to compute the threshold. Therefore, having a way of predicting the threshold by extrapolation would be desired i.e. having an explicit relation between ARL^0 and the threshold C. Intuitively, if ARL^0 is larger, more data will be taken into account implying a threshold proportionally larger. Indeed, as ARL^0 increases more data is used in the procedure and the threshold is therefore proportionally increased from how much more data is treated in the procedure. In the CUSUM case, since the alarm function is defined as the log-likelihood ratio, this results in a threshold being a log ARL^0:

- for a shift from compressed data to encrypted data:

$$C = 0.965524 \cdot \ln\left(0.030655 \cdot ARL^0 + 0.494603\right)$$

- for a shift from encrypted data to compressed data:

$$C = 0.997767 \cdot \ln\left(0.912316 \cdot ARL^0 + 7.294950\right)$$

For Shiryaev, the threshold is a linear function of ARL^0:

- for a shift from compressed data to encrypted data:

$$C = 0.647578 \cdot ARL^0 - 0.726563$$

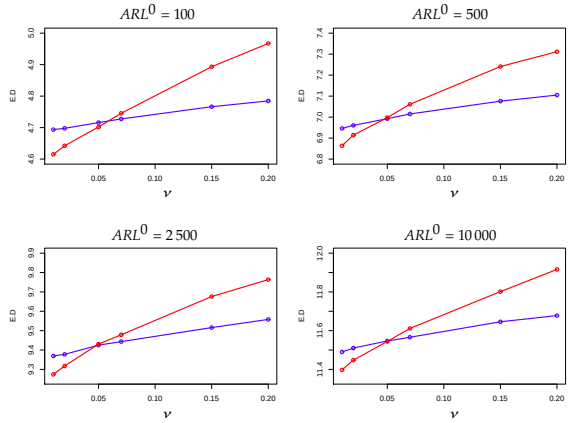

Figure 4.
Expected delays ED for a shift from encrypted to compressed data for the CUSUM procedure (blue) and Shiryaev procedure (red)

ν	Methods							
	CUSUM				Shiryaev			
	100	500	2 500	10 000	100	500	2 500	10 000
0.20	0.8950	0.9862	0.9984	0.9997	0.9859	0.9984	0.9998	1.0000
0.15	0.8727	0.9805	0.9974	0.9995	0.9736	0.9967	0.9996	0.9999
0.07	0.8031	0.9604	0.9933	0.9986	0.9147	0.9852	0.9976	0.9995
0.05	0.7634	0.9482	0.9907	0.9978	0.8708	0.9754	0.9957	0.9991
0.02	0.6171	0.8942	0.9784	0.9944	0.6927	0.9235	0.9847	0.9964
0.01	0.4712	0.8207	0.9590	0.9890	0.5153	0.8463	0.9661	0.9918

Table 5.
Predictive value $PV(\nu) = P_\nu(\theta < \tau)$, i.e. the probability that a shift has occurred when an alarm is signalled, for the CUSUM and the Shiryaev methods, for values of $ARL^0 = 100, 500, 2\,500, 10\,000$ and with different values of the parameter ν in the geometric distribution of the change-points.

- for a shift from encrypted data to compressed data:

$$C = 0.444214 \cdot ARL^0 + 0.294281$$

4. Conclusion

Using the change-point statistical process, a method to detect encrypted data in HDD was successively designed. This method is using the fact that encrypted data is uniformly distributed as opposed to other types of files. The method was designed to detect even a change with the closest files to encrypted files which are compressed data. As this method even detects a small change in the data, any bigger change will be even easier detected. Therefore this process is likely to detect encrypted data among any type of data.

Quick and accurate detection of a change is commonly the desired property of change-point detection methods. In many applications such as medicine, finance, environmental science etc., time aspects of the methods are a matter

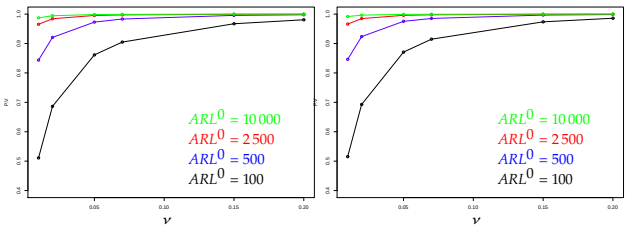

Figure 5.
Predictive values for a shift from compressed to encrypted data for the CUSUM procedure (left) and for the Shiryaev procedure (right)

of interest, e.g. expected delay in detection of a shift or probability of detecting a shift within a specified time interval. Here, however, this time aspect is not of primary interest since the data remain the same during the whole process. Here the need is to detect correctly recognized encrypted data. Therefore the probability of correctly detecting encrypted data is more relevant here. This probability shows that the method detects more than 96% of the encrypted data which is good and by extending the intervals, the method detects more than 99% of the encrypted data. By assuming that the change-points are not too close – which is a plausible assumption since it is unlikely that files are so small if the device is not too fragmented – then the method, by adding a little margin to the intervals, quickly detects 100% of the encrypted data.

The Shiryaev method turns out to be slightly better in the more important respects compared to the CUSUM method. Although the expected delay ED is bigger than CUSUM for large values of the parameter ν in the distribution of the change points, it is smaller for small values of ν which is the most relevant case for detecting encrypted data in an HDD. The Shiryaev method also detects more encrypted data than the CUSUM method and has a slightly higher predictive value PV.

All in all, this means that both methods designed with the suggested modelling, perform very well with a slight preference to the Shiryaev method for detecting encrypted data in an HDD.

To summarize, a thorough comparison between the proposed method and the aforementioned methods [5, 6, 10, 14, 25] for the situation with streamed data would be the obvious next step in this research. Also other methods, potentially building on the Kolmogorov-Smirnov statistic or the Shannon entropy and by using other anomaly detection of machine learning could be interesting candidates in such a race.

Acknowledgments

The authors wish to express their gratitude to Mattias Weckstén at Halmstad for good ideas and previous readings of the manuscript University and to Linus Nissi (previously Linus Barkman) at the Police Department of Southern Sweden for earlier work in the area.

Conflict of interest

The authors declare no conflict of interest.

Nomenclature

ARL^0	Average runlength in control
ARL^1	Average runlength out of control
ED	Expected delay of motivated alarm
CED	Conditionala expected delay of motivated alarm given that the change occurred at a specified time-point
PV	Predictive value

Author details

Eric Järpe[1] and Quentin Gouchet[2]

1 Halmstad University, P.O. Box 823, 30118 Halmstad, Sweden

2 Atsec information security, Austin, Texas, USA

*Address all correspondence to: eric.jarpe@hh.se

References

[1] Andersson, F, Nelander Hedqvist, K, Ring, J and Skarp, A. It-inslag i brottsligheten och rättsväsendets förmåga att hantera dem, Brottsförebyggande rådet, report 2016;17:1–152 [Internet]. Available from: https://bra.se/publikationer/arkiv/publikationer/2016-09-30-it-inslag-i-brottsligheten-och-rattsvasendets-formaga-att-hantera-dem.html [Accessed: 2021-12-29]

[2] Barkman, L. Detektering av krypterade filer [thesis]. diva2:428544: Halmstad University; 2011.

[3] Bassham, LE, Rukhin, A, Soto, J, Nechvatal, J, Smid, M, Barker, E, Leigh, S, Levenson, M, Vangel, M, Banks, DL, Heckert, A, Dray, J and Vo, S. A statistical test suite for random and pseudorandom number generators for cryptographic applications, National Institute of Standards and Technology, Special publication 800-22 [Internet]. 2010. Available from https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=906762 [Accessed: 2021-12-29]

[4] Bischoff, P. Best Disk Encryption Software – the Top 5 Tools to Secure Your Data, Comparitech, May 13 [Internet]. 2020. Avaialble from: https://www.comparitech.com/blog/information-security/disk-encryption-software [Accessed: 2021-12-29]

[5] Casino, F, Choo, KKR and Patsakis, C. HEDGE: Efficient Traffic Classification of Encrypted and Compressed Packets. IEEE Transactions on Information Forensics and Security. 2019;14(11):2916–2926. DOI: 10.1109/TIFS.2019.2911156

[6] Choudhury, P, Kumar, KR, Nandi, S and Athithan, G. An empirical approach towards characterization of encrypted and unencrypted VoIP traffic. Multimedia Tools and Applications. 2020;79:603–631. DOI: 10.1007/s11042-019-08088-w

[7] Frisén, M. Properties and use of the Shewhart method and its followers. Sequential Analysis. 2007;26(2):171–193. DOI: 10.1080/07474940701247164

[8] Frisén, M. Statistical surveillance. Optimality and methods. International Statistical Review. 2003;71(2):403–434. DOI: 10.1111/j.1751-5823.2003.tb00205.x

[9] Frisén, M and de Maré, J. Optimal Surveillance. Biometrika. 1991;78(2):271–280. DOI: 10.2307/2337252

[10] Hahn, D, Apthorpe, N and Feamster, N. Detecting Compressed Cleartext Traffic from Consumer Internet of Things Devices [Internet]. 2018. Available from: https://arxiv.org/abs/1805.02722 [Accessed: 2021-12-29]

[11] Järpe, E and Wessman, P. Some power aspects of methods for detecting different shifts in the mean. Communications in Statistics, Computation and Simulation. 2000;29(2):633–646. DOI: 10.1080/03610910008813632

[12] Järpe, E. Surveillance, Environmental. In: El-Shaarawi, AH and Piegorsch, WA, editors. Encyclopedia of Environmetrics. 2nd ed. Chichester, Wiley; 2013. p. 2150–2153. DOI: 10.1002/9780470057339.vas065.pub2

[13] Kozachok, AV and Spirin, AA. Model of pseudo-random sequences generated by encryption and compression algorithms. Programming and Computer Software. 2021;47(4):249–260. DOI:

10.1134/S0361768821040058

[14] Li, Y and Lu, Y. ETCC: Encrypted two-label classification using CNN. Security and Communication Networks. 2021;2021;1–11. DOI: 10.1155/2021/6633250

[15] Malhotra, P. Detection of encrypted streams for egress monitoring [thesis]. UMI Microform 1447482: Iowa State University; 2007.

[16] Page, ES. Continuous Inspection Schemes, Biometrika. 1954;41(1/2):100–115. DOI: 10.1093/biomet/41.1-2.100

[17] Paninski, L. Estimating entropy on *m* bins given fewer than *m* samples, IEEE Transactions on Information Theory. 2004;50(9):2200–2203. DOI: 10.1109/TIT.2004.833360

[18] Pearson, K. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling, Philosophical Magazine. 1900;50(302):157–175. DOI: 10.1080/14786440009463897

[19] Razali, NM and Wah, YB. Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests, Journal of Statistical Modeling and Analytics. 2011;2(1):21–33 [Internet]. Available from: https://www.researchgate.net/publication/267205556_Power_Comparisons_of_Shapiro-Wilk_Kolmogorov-Smirnov_Lilliefors_and_Anderson-Darling_Tests [Accessed: 2021-12-29]

[20] Shiryaev, AN. On Optimum Methods in Quickest Detection Problems, Theory of Probability and Its Applications. 1963;8(1):22–46. DOI: 10.1137/1108002

[21] Swedish Civil Contingencies Agency. Informationssäkerhet – trender 2015, Myndigheten för Samhällsskydd och Beredskap [Internet]. 2015. Available from: https://www.msb.se/sv/publikationer/informationssakerhet–trender-2015 [Accessed: 2021-12-29]

[22] The Swedish Police. Polisens rapport om organiserad brottslighet 2015, National operations depertment [Internet]. 2015. Available from: https://docplayer.se/844230-Polisens-rapport-om-organiserad-brottslighet-2015-polismyndigheten-nationella-operativa-avdelningen-maj-2015.html [Accessed: 2021-12-29]

[23] Westfeld, A and Pfitzmann, A. Attacks on Steganographic Systems Breaking the Steganographic Utilities EzStego, Jsteg, Steganos, and S-Tools–and Some Lessons Learned. In: Information Hiding, Third International Workshop IH'99, September–October Dresden. Germany: Springer; 2000. p. 61–76

[24] Walker, J. Pseudorandom number sequence test program, Fourmilab.ch [Internet]. 2008. Available from: https://www.fourmilab.ch/random [Accessed: 2021-12-29]

[25] Yao, Z, Ge, J, Wu, Y, Lin, X, He, R and Ma, Y. Encrypted traffic classification based on Gaussian mixture models and hidden Markov models, Journal of Network and Computer Applications. 2021;166:1–13. DOI: 10.1016/j.jnca.2020.102711