Detecting encrypted code

Järpe, E. and Wecksten, M.

September 19, 2013

Abstract

When a computer hard drive is confiscated in the conduct of a police investigation the information contained is commonly encrypted at such a level that brute force thechniques have to be used. This is too time consuming if the hard drive is large, thus demanding methods to make a smart distinction between likely encrypted and non-encrypted code in order to use the cracking resources efficiently. The methods are stopping rules according to the cutting edge of on-line change-point detection. The performance of the methods is evaluated in terms of conditional expected delay and predictive value having calibrated the methods so that they all have the same expected time till false alarm.

1 Introduction

In modern IT forensics there is a challenge to crack an encrypted hard drive that has been confiscated in the conduct of a police investigation. The strength of the contemporary tools for such protection (e.g. True crypt) is at such a level that password guessing (so-called *brute force* techniques) have to be used. This may not be a problem if there is only one or a few blocks of code are cracked within reasonable time.

However, the problem of cracking the encryption of computer code may be further complicated if the contents of the hard drive have been deleted by removing the pointers of the file allocation tables (so-called *quick deletion*). Then the knowledge of what the formats of the contents are is not present. The hard drive contents is merely a (possibly immense) sequence of blocks of code that have to be attempted for cracking. Then it may well be completely impossible to apply brute force technique on each block simply because there are so many blocks and each password guessing procedure takes too much time.

To this end a sieve discriminating between the blocks of most likely not encrypted computer code and the blocks of suspiciously encrypted (and therefore more likely harmful) computer code is desired. After having picked out the encrypted blocks one may proceed with brute force cracking in an efficient manner according to some standard. A property of a crypto which is commonly regarded as an indicator of quality is how evenly all characters of the encryption alphabet is used. Thus, a measure for telling encryption likelihood apart, is to quantify how uniformly the encryption characters are distributed; in encrypted code the characters are more likely to be evenly distributed than in text files, images, programs etc. One way of measuring how evenly the characters are distributed is by considering the character frequencies and expressing them by using the Pearson chi square statistic (see e.g. Cox and Hinkley [1]). But of course one may also consider other aspects of even distribution of the characters, such that they should not occur more clustered or regularly than what would have been the case in a completely random pattern.

This paper is all about development of such a procedure for distinguishing between the code blocks which are less likely and those which are more likely encrypted.

Previous work... In Section 2...

2 Model

Observations of blocks of code c_1, c_2, c_3, \ldots , are made consecutively at timepoints $t = 1, 2, 3, \ldots$. Some of these clusters may be subject to encryption. Depending on the degree of fragmentation of the hard drive they may be more likely to occur consecutively or not. In order to avoid tedious brute force cracking all of the clusters, a method for separating the clusters that are more likely encrypted from the clusters which are less likely encrypted is desired.

Under the assumption that the characters constituting encrypted code are more evenly distributed than the characters constituting clear text code, the Barkman crypto indicator U_t (which is nothing but the well-known Pearson chi-squared statistic, see e.g. Cox and Hinkley [1], in the special case with uniform distribution under the alternative hypothesis), may be used to distinguish code which is more likely encrypted. Simply calculating this statistic for all the clusters of the hard drive, and then brute force cracking those with an alarming score of the Barkman crypto indicator is a possibility if the hard drive is not too large.

If the hard drive is very large (it may of course not be only one physical unit but rather a composition of many units and thus, possibly, *very* large) it can be a practically impossible task to evaluate the Barkman crypto indicator for all clusters. However, if the contents is likely to be little fragmented, and accordingly clusters of the same files are more likely to be occurring consecutively, then a monitoring algorithm, applied to the on-line calculation of Barkman crypto indicator, can facilitate the search for a larger sequence of consecutive clusters that are more likely encrypted. Then the situation is more formally described as follows. Cluster observations, C_1, C_2, C_3, \ldots , are to be made on-line, i.e. one by one. Now, assuming that the code up to some random time-point $t = \tau - 1$ is not encrypted and that the code from time $t = \tau$ and on wards is, the distribution of the characters constituting C_1, C_2, C_2 $\ldots, C_{\tau-1}$ are distributed in some other way than the characters of $C_{\tau}, C_{\tau+1}$, ... The problem is then to determine that the change from not encrypted to encrypted code has occurred, as quickly after it has really happened and as accurately (i.e. with as little delay) as possible.

3 Methods

For each block c_t , where $t = 1, 2, 3, \ldots$, the number N_t of characters and the number K_t of distinct character kinds occurring in that block is counted. For

1	2	1	1	1	0	1	0	3	3
0	3	1	1	3	2	0	2	0	3
0	2	2	0	1	0	0	0	3	0
1	0	3	0	1	2	2	2	1	1
2	1	3	3	1	0	2	1	3	3
3	0	1	1	1	0	1	3	3	2

Figure 1: Code consisting of 60 characters and 4 character kinds (i.e. 0, 1, 2 and 3). Thus if this is cluster t, then $N_t = 60$ and $K_t = 4$ in this case.

all characters, count the observed frequencies $o_{i,1}$ (i.e. the number of occurrences of character 1), $o_{t,2}$ (i.e. the number of occurrences of character 2), ..., o_{t,K_t} (i.e. the number of occurrences of character K_t). From these observed frequencies the value of the *Barkman crypto indicator* (i.e. the Pearson chisquare statistic in the special case when the distribution under the alternative hypothesis is uniform)

$$U_t = \sum_{i=1}^{K_t} \frac{(K_t O_{t,i} - N_t)^2}{K_t N_t}$$
(1)

According to the Central Limit theorem, U_t is approximately χ^2 distributed when K_t and N_t are large. Thus, if we consider the problem of monitoring the degree of fit to a uniform distribution (indicating how suspiciously the code is to be encrypted) based on an on-line sequence of observations of the Barkman crypto indicator U_1, U_2, U_3, \ldots would approximatively satisfy

$$U_t \in \begin{cases} \psi^2(K_t - 1) & \text{om } t < \theta\\ \chi^2(K_t - 1) & \text{om } t \ge \theta \end{cases}$$

where $\chi^2(K_t - 1)$ is the chi square distribution with $K_t - 1$ och ψ^2 is the distribution of cX where $X \in \chi^2$ for some c > 1.

At each time t the Barkman crypto indicator, U_t , is calculated and fed to a stopping rule which is an algorithm which can be defined as

$$T = \min\{t \ge 1 : a(U_1, U_2, \dots, U_t) > C\}.$$

Let us denote $a(U_1, U_2, \ldots, U_t)$ by a_t for short. Then the stopping rule T may be described more extensively as an algorithm by the following.

Algorithm 1 For a certain threshold, $C \in \mathbb{R}$, and alarm function, a (with a convenient number of arguments), the stopping rule is:

- 1. Start: $a_1 = U_1$.
- 2. If $a_1 > C$, then we stop immediately: Goto 10. else qoto 3.
- 3. At time $t \geq 2$: make a new observation U_t .
- 4. If $a_t > C$, then we stop: Goto 10. else increase t by 1 and goto 3.
- 5. Stop and return the stopping time: T = t.

Choosing the alarm function a differently renders different kinds of stopping rules. Changing the threshold C will give the stopping rule different properties. High values of C will make the stopping rule more conservative and less likely to stop, thus giving few false alarm, but also long expected delay of motivated alarm, while lower values gives a more sensitive stopping rule with shorter expected delay, but more likely to give false alarm.

4 Sufficient statistic

5 The change-point problem

6 Results

6.1 Stopping rules

In this paper we consider the Shewhart, CUSUM, Shiryaev and Roberts methods. The *Shewhart method* (Shewhart [9]) is the stopping rule

$$T_W = \min\{t \ge 1 : U_t > C\}.$$

The CUSUM method (Page [7]) is

$$T_C = \min\{t \ge 1 : a_t > C\}$$

where

$$a_t = \begin{cases} 0 & \text{if } t = 0\\ \max(0, a_{t-1}) + n \ln c - \frac{c-1}{2c} U_t & \text{if } t = 1, 2, 3, \dots \end{cases}$$

The Shiryaev method (Shiryaev [10]) is

$$T_Y = \min\{t \ge 1 : a_t > C\}$$

where

$$a_t = \begin{cases} 0 & \text{if } t = 0\\ (1 + a_{t-1}) \exp\left(n \ln c - \frac{c-1}{2c} U_t\right) & \text{if } t = 1, 2, 3, \dots \end{cases}$$

6.2 Evaluation

In order to evaluate the stopping rules theoretically the threshold is chosen so that the $ARL^0 = E(T | T < \theta) = 100$ (i.e. the expected time until there is a false alarm is 100). Then aspects of motivated alarm are calculated and compared. All calculations here are based on simulations. The samplesize in the simulations of the ARL^0 is 100 000 observations for each method and size of shift c = 1.1, c = 1.2 and c = 1.3.

The first aspect of motivated alarm to be examined is the *conditional* expected delay, $CED(t) = E(T - \theta | T \ge \theta = t)$. This was simulated with a samplesize of 1 000 000 observations for each point in the plot of Figure 3. This performance measure indicates how quickly a method signals alarm after the change has happened.

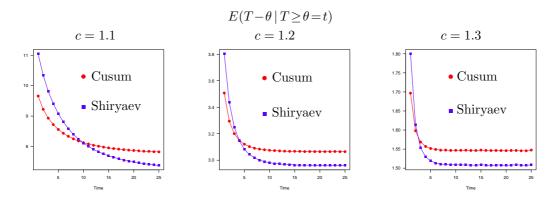


Figure 2: Conditional expected delay for the Cusum (red lines and circles) and Shiryaev (blue lines and squares) methods for shift size c = 1.1 (left plot), c = 1.2 (middle plot) and c = 1.3 (right plot).

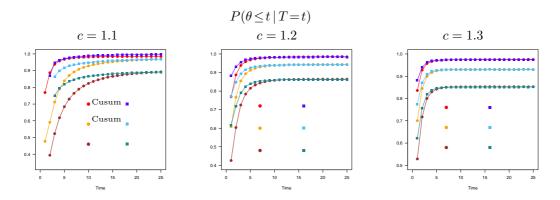


Figure 3: Predictive value for the Cusum (red, orange and brown lines and circles) and Shiryaev (blue, light blue and turqoise lines and squares) methods for shift size c = 1.1 (left plot), c = 1.2 (middle plot) and c = 1.3 (right plot).

The second aspect of motivated alarm is *predictive value*, $PV(t,\nu) = P(\theta \le t | T = t)$ where the change-point θ is assumed to be geometrically distrubuted with parameter ν with the interpretation $\mu = P(\theta = t | \theta > t-1)$. In each of the plots the sample size was 10 000 000 observations for each method. The predictive value shows how much credibility one should assert to a method in the case when the change-point occurs at time t.

6.3 Examples

7 Discussion

References

- [1] D. COX AND D.V. HINKLEY (1974) *Theoretical Statistics*, Chapman & Hall.
- [2] M. FRISÉN (2003) Statistical surveillance. Optimality and methods. Int. Stat. Rev., 71 (2), 403–434.
- [3] M. FRISÉN AND P. WESSMAN (1999) Evaluations of likelihood ratio methods for surveillance: differences and robustness. *Comm. Statist. Simulation Comput.*, 28 (3), 597–622.
- [4] E. JÄRPE (1999) Surveillance of the interaction parameter in the Ising model. Comm. Statist. Theory Methods, 28 (12), 3009–3027.
- [5] E. JÄRPE (2002) Surveillance, environmental. In: *Encyclopedia of Envi*ronmetrics (eds. A.H. El-Shaarawi and W.W. Piegorsch), Wiley. 2150– 2153.
- [6] E. JÄRPE AND P. WESSMAN (2000) Some power aspects of methods for detecting different shifts in the mean. *Comm. Statist. Simulation Comput.*, **29** (2), 633–646.
- [7] E.S. PAGE (1954) Continuous inspection schemes. Biometrika, 41, 100– 115.
- [8] S.W. ROBERTS (1959) Control chart tests based on geometric moving averages. *Technometrics*, **42** (1), 239–250.
- [9] W.A. SHEWHART (1931), Economic Control of Quality Control, *Reinhold Company*, Princeton N.J.

- [10] A.N. SHIRYAEV (1963) On Optimum Methods in Quickest Detection Problems. Theory Probab. Appl., 8 (1), 22–46.
- [11] M.S. SRIVASTAVA AND Y. WU (1993), Comparison of some EWMA, CUSUM and Shiryaev-Roberts procedures for detecting a shift in the mean, Ann. Statist., 21 (2), 645–670.