A PROBABILISTIC MODEL TO EVALUATE COMPUTER SOFTWARE PIRACY

R.J. Whitley[(1)] and T.V. Hromadka II[(2)]

Abstract

The growing occurrence of computer software piracy has led to a new area of research, i.e., the development of methods to be used to supply evidence that software was copied.

One method to argue that computer source code was copied is to examine the occurrence of strings of binary code (ones and zeroes) between the alleged parent and pirate codes. Given the occurrence of a lengthy identical string between codes, and that string represents a development of executable code (versus data blocks that can be argued to exist in only one fashion), a model of the probability of repetition of such a string of code occurring between so-called independently derived source codes can be formulated. The developed probabilistic results can also be approximated by a simpler formula derived herein. A computer program and example computations are presented.

---

(1) Professor, Department of Mathematics, University of California, Irvine, CA 92717

(2) Associate Professor, Department of Mathematics, California State University, Fullerton, CA 92634

A probabilistic model to evaluate computer software piracy

R. J. WHITLEY

Department of Mathematics, University of California, Irvine, CA 92717, U.S.A.

T. V. HROMADKA II

Department of Mathematics, California State University, Fullerton, CA 92634, U.S.A.

The growing occurrence of computer software piracy has led to a new area of research, i.e., the development of methods to be used to supply evidence that software was copied.

One method to argue that computer source code was copied is to examine the occurence of strings of binary code (ones and zeroes) between the alleged parent and pirate codes. Given the occurrence of a lengthy identical string between codes, and that string represents a development of executable code (versus data blocks that can be argued to exist in only one fashion), a model of the probability of repetition of such a string of code occurring between so-called independently derived source codes can be formulated. The developed probabilistic results can also be approximated by a simpler formula derived herein. A computer program and example computations are presented.

INTRODUCTION

The growing occurrence of computer software piracy has led to a new area of research, i.e., the development of methods to be used to supply evidence that software was copied. This problem is difficult due to the argument that the parent software was developed by a knowledgeable person and hence the resulting code should be a more probable outcome than another statement of code. Another argument in defense is that there can only be a finite number of ways to write a segment of code.

One method to argue code was copied is to examine the occurrence of strings of binary code (ones and zeroes) between the alleged parent and pirate codes. Given the occurrence of a lengthy identical string between codes, and that string represents a development of executable code (versus data blocks that can be argued to exist in only one fashion), a model of the probability of repetition of such a string of code occurring between so-called independently derived source codes can be formulated. The probabilistic model presented herein assumes independence and equal probability of the string occurring within a total source code string size. Although an argument can be formulated that a string of binary code is not composed of independent trials of a Bernoulli random variable (i.e., the ones and zeroes are independent) due to the structure of the language, this argument can be accommodated by modeling a string as a set of independent events rather than independent binary digits; consequently, a large string may be actually considered to be a string of independent blocks of binary code of a finite size, and thus reduce the sample size of the string and also the size of the total source code.

In this paper, the general model is developed. Extension of the probabilistic model to handle the cases of binary code blocks, or different probability weightings is straight forward from the supplied development.

The specific probabilistic problem is: For a given probability, how large must the total source code be in order to experience a specific binary code string? In the model development, the analog is made to a simple fair coin toss experiment, where heads (H) has the probability of a specific binary code string. A simple approximation formula to the probabilistic model is developed, and a computer program presented.

PROBABILISTIC MODEL

Consider tossing a fair coin. For a number m let p_n denote the probability that in n tosses there will be at least one run of heads of length m. To compute this probability is the same as computing the number H_n of different outcomes of n tosses which contain at least one run of m heads, since[1]

$$p_n = (1/2^n)H_n \tag{1}$$

It is useful to also consider the set E_n of all outcomes of n tosses in which at least one run of m heads occurs; E_n has H_n members. The integer m is regarded as fixed,

Paper accepted January 1991. Discussion closes January 1992.

simplifying the notation which would otherwise be, say, $p_{n,m}$ and $H_{n,m}$.

A recursive formula can be derived by considering the events in E_{n+1} and looking at two cases.

Case 1 is when the event of a run of m heads has already occured in the first n trials. Letting X_i be H or T depending in the results of the i-th toss, case 1 has the form $X_1X_2 \ldots X_nX_{n+1}$ where $X_1X_2 \ldots X_n$ is an event in E_n and X_{n+1} is either is H or T, there are $2H_n$ events in case 1.

Case 2 is the event, disjoint from case 1, in which there is no run of heads of length m in the first n trials but there is such a run in $n+1$ trials. For this to occur, the event must have the form

$$X_1X_2 \ldots X_{n-m}\ THHH \ldots H \tag{2}$$

where the sequence of the last $m+1$ outcomes begins with a T and is followed by m Hs. Also, for the outcome in equation (2) to belong to case 2 there cannot be a run of m heads in the entries $X_1X_2 \ldots X_{n-m}$; i.e. $X_1X_2 \ldots X_{n-m}$ does not belong to E_{n-m}. Thus, there are

$$2^{n-m} - H_{n-m} \tag{3}$$

different events in case 2.

The set E_{n+1} consists of the disjoint union of case 1 and case 2, so

$$H_{n+1} = 2H_n + 2^{n-m} - H_{n-m} \tag{4}$$

Dividing by 2^{n+1} and using equation (1)

$$p_{n+1} = p_n + \frac{(1 - p_{n-m})}{2^{m+1}} \tag{5}$$

This recursively defines, p_{n+1}, beginning with

$$p_j = 0, j < m \text{ and } p_m = 1/2^m \tag{6}$$

For large m, some ideas of the size of the numbers involved can be obtained by considering another recursively defined sequence. Consider α_n defined by

$$\alpha_j = p_j; j = 0, 1, \ldots, 2m-1 \tag{7}$$

$$\alpha_{n+1} = \alpha_n + 1/2^{m+1} \tag{8}$$

Since

$$p_{2m-1} = (m+1)/2^{m+1} \tag{9}$$

$$\alpha_n = (n - m + 2)/2^{m+1} \text{ for } n \geq 2m \tag{10}$$

By construction,

$$\alpha_n \geq p_n \text{ for all } n \tag{11}$$

Given a value c, $0 < c < 1$, the value of n such that

$$\alpha_n \geq c \tag{12}$$

is

$$n = c2^{m+1} + m - 2 \tag{13}$$

So n will have to be at least this large, for $n \geq 2m$, to have $p_n \geq c$. For example, suppose you want to take n large enough so that the probability of 100 consecutive heads (or a string of size 100) will be .5; the smallest n that will do will be at least as large as the n for which $q_n \geq .5$, i.e. $n = .5(2^{101}) + 98 = 1.27 \times 10^{30}$.

It is possible to obtain a very accurate computable approximate formula for [p_n by arguing as follows: with the change of variable

$$q_n = 1 - p_n \tag{14}$$

equation (5) becomes the homogeneous difference equation

$$q_{n+1} = q_n - (q_{n-m}/2^{m+1}) \tag{15}$$

Substituting $q_n = x^n$ in this equation gives the related equation

$$f(x) = x^{m+1} - x^m + 1/2^{m+1} = 0 \tag{16}$$

The solution q_n is a linear combination of n-th powers of roots of equation (16); for an approximate formula it is hoped that for large n the largest root will give the dominant term.

To approximate the largest root of equation (16), note that

$$f(1) = 1/2^{m+1} \tag{17}$$

is quite small for large m and so $x_0 = 1$ would be a good initial guess at this root. Apply Newton's method for a better guess

$$x_1 = x_0 - f(x_0)/f'(x_0) = 1 - 1/2^{m+1} \tag{18}$$

This leads to the approximation

$$\hat{p}_n = 1 - (1 - 1/2^{m+1})^n \tag{19}$$

This approximation does not satisfy any of the boundary conditions of equation (6), but, as we will see below, the derived formula (20) is quite accurate.

Given $0 < c < 1$, the first value of n for which $\hat{p}_n \geq c$ is

$$n = \ln(1-c)/\ln(1 - 1/2^{m+1}) \approx 2^{m+1}(-\ln(1-c)) \tag{20}$$

For example, for $m = 100$, and $c = .5$ $n = 1.7 \times 10^{30}$; compare this with the lower bound given above.

The table below gives for $c = .1(.1).5$ and for $n = 4(1)20$

(a) The value of the first n for which $p_n \geq c$, computed from the recursion (10).
(b) The value of n computed from the right hand side of equation (20).
(c) The lower bound of equation (13).

REFERENCE

1 Helstrom, C. W. *Probability and Stochastic Processes for Engineers*, MacMillan Publishing Co., 1984

```
m = 4
pn = 0.12500000 n = 6     approx. n = 3     bound  = 5
pn = 0.21679688 n = 9     approx. n = 7     bound  = 8
pn = 0.32421875 n = 13    approx. n = 11    bound  = 11
pn = 0.41693115 n = 17    approx. n = 16    bound  = 14
pn = 0.51514530 n = 22    approx. n = 22    bound  = 17

m = 5
pn = 0.10937500 n = 10    approx. n = 7     bound  = 9
pn = 0.21021271 n = 17    approx. n = 14    bound  = 15
pn = 0.31159061 n = 25    approx. n = 23    bound  = 22
pn = 0.41017180 n = 34    approx. n = 33    bound  = 28
pn = 0.50324029 n = 44    approx. n = 44    bound  = 34

m = 6
pn = 0.10034180 n = 17    approx. n = 13    bound  = 16
pn = 0.20497159 n = 32    approx. n = 29    bound  = 29
pn = 0.30319989 n = 48    approx. n = 46    bound  = 42
pn = 0.40420780 n = 67    approx. n = 65    bound  = 55
pn = 0.50301594 n = 89    approx. n = 89    bound  = 67

m = 7
pn = 0.10260123 n = 32    approx. n = 27    bound  = 30
pn = 0.20149082 n = 61    approx. n = 57    bound  = 56
pn = 0.30083370 n = 94    approx. n = 91    bound  = 81
pn = 0.40001731 n = 132   approx. n = 131   bound  = 107
pn = 0.50144943 n = 178   approx. n = 177   bound  = 132

m = 8
pn = 0.10158655 n = 60    approx. n = 54    bound  = 57
pn = 0.20094174 n = 119   approx. n = 114   bound  = 108
pn = 0.30051364 n = 186   approx. n = 183   bound  = 159
pn = 0.40091199 n = 264   approx. n = 262   bound  = 210
pn = 0.50097240 n = 356   approx. n = 355   bound  = 261

m = 9
pn = 0.10006459 n = 114   approx. n = 108   bound  = 109
pn = 0.20046290 n = 234   approx. n = 228   bound  = 211
pn = 0.30008661 n = 369   approx. n = 365   bound  = 314
pn = 0.40044137 n = 526   approx. n = 523   bound  = 416
pn = 0.50038894 n = 711   approx. n = 710   bound  = 518

m = 10
pn = 0.10013437 n = 223   approx. n = 216   bound  = 212
pn = 0.20012739 n = 463   approx. n = 457   bound  = 417
pn = 0.30008859 n = 735   approx. n = 730   bound  = 622
pn = 0.40005301 n = 1049  approx. n = 1046  bound  = 827
pn = 0.50017293 n = 1421  approx. n = 1420  bound  = 1031

m = 11
pn = 0.10014117 n = 440   approx. n = 432   bound  = 418
pn = 0.20010925 n = 921   approx. n = 914   bound  = 828
pn = 0.30002588 n = 1466  approx. n = 1461  bound  = 1237
pn = 0.40007713 n = 2096  approx. n = 2092  bound  = 1647
pn = 0.50010228 n = 2841  approx. n = 2839  bound  = 2056
```

```
m = 12
pn = 0.10002148 n = 872     approx. n = 863    bound  = 829
pn = 0.20007727 n = 1836    approx. n = 1828   bound  = 1648
pn = 0.30004889 n = 2928    approx. n = 2922   bound  = 2467
pn = 0.40005115 n = 4189    approx. n = 4185   bound  = 3286
pn = 0.50002444 n = 5680    approx. n = 5678   bound  = 4105

m = 13
pn = 0.10001049 n = 1736    approx. n = 1726   bound  = 1649
pn = 0.20000118 n = 3664    approx. n = 3656   bound  = 3287
pn = 0.30000211 n = 5850    approx. n = 5844   bound  = 4926
pn = 0.40001930 n = 8374    approx. n = 8369   bound  = 6564
pn = 0.50002542 n = 11359    approx. n = 11357   bound  = 8202

m = 14
pn = 0.10000197 n = 3463    approx. n = 3452   bound  = 3288
pn = 0.20000648 n = 7321    approx. n = 7312   bound  = 6565
pn = 0.30001379 n = 11695    approx. n = 11688   bound  = 9842
pn = 0.40001230 n = 16744    approx. n = 16739   bound  = 13119
pn = 0.50000004 n = 22715    approx. n = 22713   bound  = 16395

m = 15
pn = 0.10000995 n = 6917    approx. n = 6905   bound  = 6566
pn = 0.20000637 n = 14634    approx. n = 14624   bound  = 13120
pn = 0.30000511 n = 23383    approx. n = 23375   bound  = 19673
pn = 0.40000411 n = 33483    approx. n = 33477   bound  = 26227
pn = 0.50000496 n = 45429    approx. n = 45426   bound  = 32780

m = 16
pn = 0.10000635 n = 13823    approx. n = 13810   bound  = 13121
pn = 0.20000494 n = 29259    approx. n = 29248   bound  = 26228
pn = 0.30000420 n = 46759    approx. n = 46750   bound  = 39335
pn = 0.40000225 n = 66961    approx. n = 66955   bound  = 52442
pn = 0.50000095 n = 90855    approx. n = 90852   bound  = 65549

m = 17
pn = 0.10000074 n = 27633    approx. n = 27620   bound  = 26229
pn = 0.20000049 n = 58507    approx. n = 58496   bound  = 52443
pn = 0.30000012 n = 93509    approx. n = 93500   bound  = 78658
pn = 0.40000015 n = 133916    approx. n = 133910   bound  = 104872
pn = 0.50000144 n = 181708    approx. n = 181704   bound  = 131086

m = 18
pn = 0.10000119 n = 55254    approx. n = 55239   bound  = 52444
pn = 0.20000098 n = 117004    approx. n = 116991   bound  = 104873
pn = 0.30000027 n = 187010    approx. n = 187000   bound  = 157302
pn = 0.40000081 n = 267827    approx. n = 267820   bound  = 209731
pn = 0.50000007 n = 363412    approx. n = 363409   bound  = 262159

m = 19
pn = 0.10000047 n = 110494    approx. n = 110479   bound  = 104874
pn = 0.20000029 n = 233996    approx. n = 233983   bound  = 209732
pn = 0.30000011 n = 374011    approx. n = 374001   bound  = 314589
pn = 0.40000027 n = 535647    approx. n = 535639   bound  = 419447
pn = 0.50000001 n = 726821    approx. n = 726817   bound  = 524304

m = 20
pn = 0.10000006 n = 220973    approx. n = 220957   bound  = 209733
pn = 0.20000024 n = 467980    approx. n = 467966   bound  = 419448
pn = 0.30000025 n = 748013    approx. n = 748002   bound  = 629163
pn = 0.40000014 n = 1071287    approx. n = 1071279   bound  = 838878
pn = 0.50000005 n = 1453639    approx. n = 1453635   bound  = 1048593
```

APPENDIX B. SOURCE CODE

Computes the probability p[n] of getting at least one run of heads of input length m in n trials of a fair coin. The values of n for which p[n] first exceeds .1(.1).5 are printed. An approximate formula and a simple lower bound for n, valid for n great than or equal to 2m, are also printed.

```
uses printer;
var
     p:array[0..50] of double;
     n:longint;
     k1,k2,k3,i,m:integer;
     c,L:double;
begin
writeln('input m');
readln(m);
writeln(lst,'m = ',m);
c:=exp(-(m+1)*ln(2));
L:=0.1;
for i:=0 to m-1 do begin
     p[i]:=0;
end; {for}
p[m]:=exp(-m*ln(2));
n:=m;
repeat
     k1:=n mod (m+1);
     k2:=(n+1) mod (m+1);
     k3:=(n-m) mod (m+1);
     p[k2]:=p[k1]+c*(1-p[k3]);
     if (p[k2] >L) then
          begin
          writeln(lst,'pn = ',p[k2]:1:8,' n = ',n+1,
               '     approx. n = ',round(-ln(1-L)*exp((m+1)*ln(2))),
               '     bound  =',trunc(L*exp((m+1)*ln(2))+m-2));
          L:=L+0.1;
     end;{if}
     n:=n+1;
until (L>0.5);
writeln(lst,'');
end. {program}
```