Методика экспертизы программного обеспечения на основе усовершенствованного алгоритма расчета скользящей энтропии двоичных файлов

С.В. Карбовский

***Аннотация*—В статье рассматривается алгоритм расчета скользящей энтропии двоичного файла с пересечением между соседними блоками. Актуальность исследования обусловлена широким распространением энтропийного анализа в экспертизе программного обеспечения. Используемый в настоящее время алгоритм расчета энтропии с пересечением между соседними блоками в худшем случае имеет квадратичную сложность. Вследствие этого применяемый в инструментах анализа алгоритм расчета энтропии разделяет входной файл на непересекающиеся блоки, что понижает точность энтропийного анализа. В работе проведен анализ изменения информационной энтропии сообщения при изменении в нем одного символа. Определено, что расчеты изменения энтропии, вне зависимости от соотношения частоты появления в сообщении удаляемого и добавляемого символов, могут быть проведены за константное время. В результате анализа выявленных зависимостей разработан более производительный алгоритм расчета скользящей энтропии двоичных файлов с пересекающимися блоками. Показано, что разработанный алгоритм позволяет рассчитать энтропию с произвольной величиной смещения между соседними блоками файла. Проведена экспериментальная оценка точности и производительности алгоритма. Выявлено, что прирост производительности при использовании разработанного алгоритма увеличивается при уменьшении смещения между соседними блоками. Исследование предназначено для специалистов в области системного анализа программного обеспечения, а также в области обратного проектирования программного обеспечения.**

***Ключевые слова*— энтропийный анализ, скользящая энтропия, анализ алгоритмов, обратное проектирование.**

I. Введение

Информационная энтропия по Шеннону (далее – энтропия) – численная мера, характеризующая разброс вероятности появления в сообщении символов алфавита данного сообщения. Как известно, энтропия не зависит от взаимного расположения символов в сообщении, а только от их количественного соотношения; иными словами, энтропия не учитывает синтаксические и семантические особенности сообщения. Например, сообщение размером 256 байт, которое содержит упорядоченную последовательность байт от 0x00 до 0xFF, обладает максимально возможной восьмибитной энтропией, хотя не является ни сжатым, ни зашифрованным, ни случайным [1].

Значение энтропии также показывает среднее количество информации, передаваемой каждым символом сообщения. Данный факт объясняет высокую энтропию сжатого или зашифрованного сообщения [2]. Описанное выше свойство энтропии привело к широкому использованию энтропийного анализа в экспертизе программного обеспечения. Суть метода энтропийного анализа заключается в экспертной оценке динамики информационной энтропии двоичного файла, подвергаемого анализу. Автоматизированные инструменты анализа позволяют сделать предположение о содержимом исследуемого объекта по его информационной энтропии (метод сегментации файла [3], метод идентификации типа файла [4]). Наличие в исполняемом файле фрагмента с высокой энтропией используют как признак вредоносного файла [5, 6]. Энтропийный анализ используется и при анализе встроенного программного обеспечения (прошивок) телекоммуникационного оборудования, а также систем Интернета вещей [3, 7]. Поскольку структурированные данные могут обладать высокой энтропией [8], фрагменты прошивки с высокой энтропией имеют повышенный приоритет при проведении экспертизы. Такие фрагменты могут содержать в себе ценные сведения – ключи шифрования, данные аутентификации, конфигурационные файлы, и т.д.

Необходимым этапом энтропийного анализа двоичного файла является определение в нем динамики информационной энтропии. С этой целью при расчете энтропии используется метод скользящего окна [2]. При этом в файле выделяются непрерывные блоки, размер которых равен ширине окна; блоки при этом могут пересекаться. Стоит отметить, что использование ширины окна менее 256 символов при расчете восьмибитной энтропии целесообразно лишь в случае предположения об использовании в объекте исследования сокращенного алфавита, поскольку в таком случае изменяется область значений энтропии, что требует масштабирования результатов [9].

Полученные значения информационной энтропии каждого блока упорядочиваются в соответствии со смещением от начала файла до начала блока. Ряд значений энтропии затем может быть представлен в виде графика (пример приведен на рисунке 1.), в результате анализа которого производится экспертная оценка.

Статья получена 25 апреля 2023.

Сергей Витальевич Карбовский, ООО «Статус Комплайнс» (e-mail: kserx05@yandex.ru).

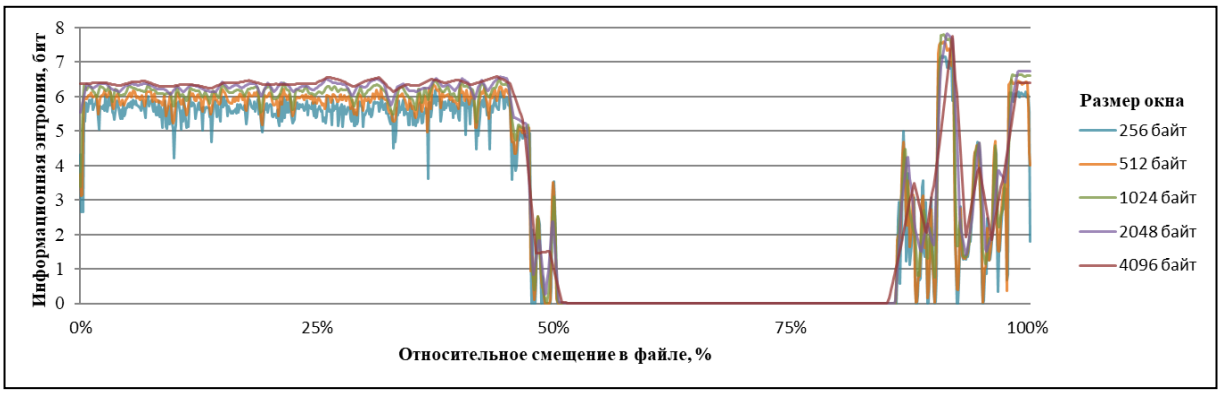

Рис. 1 – График энтропии исполняемого файла

Анализ исходного кода инструментов [11-16] показал, что в настоящее время используются две равноправные методики расчета энтропии со скользящим окном:

1) При расчете используется прямоугольное окно с шириной, равной степени 2, окна не перекрываются;

2) Файл разбивается на некоторое число блоков равного размера, при расчете используется прямоугольное окно с шириной, соответствующей размеру блока, окна не перекрываются.

Для получения данных, представленных на рисунке 1 использован исполняемый файл cmd.exe, входящий в состав массива данных PackingData, собранного в рамках исследования [10]. Расчеты проведены по упомянутой выше методике 1) с размерами окна, равными 256, 512, 1024, 2048 и 4096 байт. На рисунке 1 отчетливо видны три сегмента, соответствующие сегментам исполняемого файла.

Обнаружено, что реализованные в инструментах [11-16] алгоритмы расчета энтропии со скользящим окном используют прямоугольное окно без пересечения между соседними окнами вследствие оптимизации производительности данных алгоритмов. Тем не менее, используемое в настоящий момент алгоритмическое решение приводит к снижению точности энтропийного анализа. В частности, оно не позволяет точно определить смещение в файле, по которому обнаружен фронт или спад значения информационной энтропии. **Целью исследования** является разработка нового алгоритма расчета скользящей энтропии двоичных файлов с пересекающимися блоками, который позволяет повысить точность энтропийного анализа.

II. Материалы и методы

Информационная энтропия по Шеннону рассчитывается по формуле:

$$H = -\sum_{x:\, p(x)>0} p(x)\, log_2\, p(x)\,, x \in A, \#(1)$$

где x – символы из множества A допустимых символов сообщения (входного алфавита), а $p(x)$ – вероятность появления в сообщении соответствующего символа. Для случая восьмибитной энтропии сообщения $S = \langle s_n \rangle$ формула (1) принимает вид:

$$H(S) = -\sum_{\substack{i=0 \\ p(x)>0}}^{255} p(i)\, log_2\, p(i)\,, \#(2)$$

где $p(i)$ – вероятность появления символа i в сообщении S. Вероятность интерпретируется в классическом смысле и равна частоте появления данного символа:

$$p_i = \frac{c_i}{n}, \qquad c_i = \sum_{j=1}^{n} \begin{cases} 1, s_j = i \\ 0, s_j \neq i \end{cases}, \qquad n = \sum_{i=0}^{255} c_i,$$

где c_i – количество повторений символа i из алфавита сообщения S, а n – длина сообщения S.

Далее приведен алгоритм расчета энтропии, использующего скользящее окно со смещением между соседними блоками, реализации которого используются в [11-16]. Из записи алгоритма видно, что символы, попадающие одновременно в соседние окна, участвуют в расчете энтропии более одного раза.

$SlidingWindowEntropyOriginal(S, n, w, v)$

Вход:
сообщение $S = \langle s_n \rangle$;
ширина окна w $(w \leq n)$;
смещение между соседними окнами v.

Выход: массив значений энтропии H со скользящим окном ширины w.

$p \leftarrow \left\lceil \frac{n-w+1}{v} \right\rceil$
$H[p] \leftarrow \emptyset$
$bound_{left} \leftarrow 0$
for k ***in*** $[0, p)$
  $C[256] \leftarrow \emptyset$
  for i ***in*** $[bound_{left}, bound_{left} + w)$
    $C[S[i]] \leftarrow C[S[i]] + 1$
  end for
  $H[k] = -\sum_{\substack{j=0 \\ C[j]>0}}^{255} \frac{C[j]}{w} log_2 \left(\frac{C[j]}{w}\right)$
  $bound_{left} \leftarrow bound_{left} + v$
end for

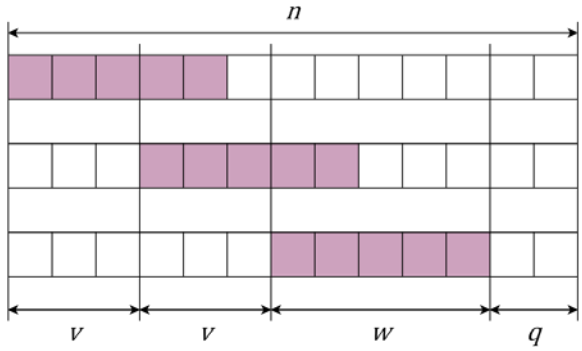

Рис. 2 – Иллюстрация метода скользящего окна

В приведенном алгоритме определено заполнение выходного массива значениями скользящей энтропии. Размер выходного массива равен $\lceil (n-w+1)/v \rceil$. Данный факт определен в результате рассуждения, проиллюстрированного на рисунке 2.

Рассуждение состоит в следующем. Через каждые v байт входного массива начинается новый блок длины w. Тогда запишем:

$$n = kv + w + q \ \#(3)$$
$$0 \le q \le v-1, \qquad k \in \mathbb{N}_0$$

Количество отрезков c в таком случае на единицу больше коэффициента k. Подставив неравенство для q в формулу 3, получим:

$$\frac{n-w+1}{v} \le c \le \frac{n-w}{v} + 1$$

Требование $k \in \mathbb{N}_0$ определяет количество блоков:

$$c = \left\lceil \frac{n-w+1}{v} \right\rceil = \left\lfloor \frac{n-w}{v} + 1 \right\rfloor \ \#(4)$$

Из формулы 4 следует, что в процессе выполнения алгоритма операция доступа к символам сообщения S выполняется $w\left\lfloor \frac{n-w}{v} + 1 \right\rfloor$ раз. Таким образом, вычислительная сложность алгоритма равна $O\left(\frac{w(n-w)}{v}\right)$. Вычислительная сложность в частных случаях с указанием условий приведена в таблице 1.

Таблица 1 – Вычислительная сложность алгоритма расчета энтропии

Случай	Условие	Вычислительная сложность
Лучший	$n = w$	$O(1)$
Средний	$w = const$ $v = const$	$O(n)$
Худший	$w = \beta n,$ $\beta \in \mathbb{R}: [0;1)$ $v = const$	$O(n^2)$

Из таблицы 1 видно, что в худшем случае сложность алгоритма является квадратичной. Данному случаю соответствует разделение входных данных на некоторое число блоков равного размера (например, в [11, 14]); при этом значения энтропии блоков сглажены за счет вычисления энтропии в промежуточных точках. Квадратичная сложность алгоритма объясняется тем, что символы, попадающие одновременно в соседние окна, участвуют в расчете энтропии до $\beta n/w$ раз. Для оптимизации данного алгоритма в худшем случае достаточно выполнить следующее условие: *каждый символ сообщения обрабатывается алгоритмом* $O(1)$ *раз*.

В ходе исследования с целью проверки выполнимости указанного условия проведен анализ изменения энтропии сообщения при изменении в нем одного символа (что эквивалентно сдвигу скользящего окна на один символ).

III. Результаты

Рассмотрим добавление к сообщению S одного символа, равного k. Энтропия полученного сообщения рассчитывается по формуле:

$$H(S.k) = log_2(n+1) - \frac{1}{n+1}\left(\sum_{\substack{i=0 \\ i \ne k \\ c_i>0}}^{255} c_i \, log_2(c_i) + (c_k+1)\, log_2(c_k+1)\right)$$

Алгебраические преобразования дают формулы, представленные в таблице 2.

Таблица 2 – Изменение энтропии сообщения в результате добавления символа

Количество повторений добавляемого символа	Энтропия сообщения
$c_k = 0$	$H(S.k) = \frac{n}{n+1}H(S) + \frac{(n+1)\,log_2(n+1) - n\,log_2(n)}{n+1}$
$c_k > 0$	$H(S.k) = \frac{n}{n+1}H(S) + \frac{(n+1)\,log_2(n+1) - n\,log_2(n)}{n+1} - \frac{(c_k+1)\,log_2(c_k+1) - c_k\,log_2(c_k)}{n+1}$

Рассмотрим удаление из сообщения S одного символа, равного t. Энтропия полученного сообщения рассчитывается по формуле:

$$H(S/t) = log_2(n-1) - \frac{1}{n-1}\left(\sum_{\substack{i=0 \\ i \ne t \\ c_i>0}}^{255} c_i \, log_2(c_i) + (c_t-1)\, log_2(c_t-1)\right)$$

Формулы для энтропии при удалении символа представлены в таблице 3.

Таблица 3 – Изменение энтропии сообщения в результате удаления символа

Количество повторений удаляемого символа	Энтропия сообщения
$c_t = 1$	$H(S\backslash t) = \frac{n}{n-1}H(S) - \frac{n\,log_2(n) - (n-1)\,log_2(n-1)}{n-1}$
$c_t > 1$	$H(S\backslash t) = \frac{n}{n-1}H(S) - \frac{n\,log_2(n) - (n-1)\,log_2(n-1)}{n-1} + \frac{c_t\,log_2(c_t) - (c_t-1)\,log_2(c_t-1)}{n-1}$

Рассмотрим теперь случай замены в сообщении S

одного символа, равного t, на символ, равный k. Операция замены символа является комбинацией операций удаления символа t и добавления символа k, при этом длина полученного сообщения не изменяется. В следующих вырожденных случаях энтропия после замены символа не изменяется:

- $t = k$ (некоторый символ заменяется на такой же символ);
- $c_t = c_k + 1$ (вклад, который заменяемые символы суммарно вносят в энтропию, после замены не изменяется).

Формулы для энтропии при замене одного символа, полученные на основе таблиц 2 и 3, представлены в таблице 4.

Таблица 4 – Изменение энтропии сообщения в результате замены символа

Количество повторений добавляемого и удаляемого символов	Энтропия сообщения
$t = k$	$H(S.k\backslash t) = H(S)$
$t \neq k,$ $c_t = c_k + 1$	$H(S.k\backslash t) = H(S)$
$t \neq k,$ $c_k = 0,$ $c_t > 1$	$H(S.k\backslash t) = H(S) + \frac{c_t \log_2(c_t) - (c_t - 1)\log_2(c_t - 1)}{n}$
$t \neq k,$ $c_k > 0,$ $c_t = 1$	$H(S.k\backslash t) = H(S) - \frac{(c_k + 1)\log_2(c_k + 1) - c_k \log_2(c_k)}{n}$
$t \neq k,$ $c_k > 0,$ $c_t > 1$	$H(S.k\backslash t) = \frac{c_t \log_2(c_t) - (c_t - 1)\log_2(c_t - 1)}{n}$ $- \frac{(c_k + 1)\log_2(c_k + 1) - c_k \log_2(c_k)}{n} + H(S)$

Формулы, приведенные в таблице 3, позволяют записать алгоритм расчета энтропии файла со скользящим окном. Алгоритм включает в себя следующие шаги:

1) Расчет значения энтропии для первого блока, выделенного во входном файле;
2) Замена в блоке первого символа на следующий за блоком;
3) Расчет изменения энтропии;
4) Проверка конца файла;
5) Завершение алгоритма или переход к шагу 2.

Далее приведена запись алгоритма на псевдоязыке.

$SlidingWindowEntropyModified(S, n, w, v)$

Вход:
сообщение $S = \langle s_n \rangle$;
ширина окна w ($w \leq n$);
смещение между соседними окнами v.

Выход: массив значений энтропии H со скользящим окном ширины w.

$p \leftarrow \left\lceil \frac{n-w+1}{v} \right\rceil$
$H[p] \leftarrow \emptyset$
$p_{max} \leftarrow n - w + 1$
$C[256] \leftarrow \emptyset$
$H_{cur} \leftarrow 0$

for i ***in*** $[0, w)$
  $C[S[i]] \leftarrow C[S[i]] + 1$
end for
$H_{cur} \leftarrow -\sum_{\substack{j=0 \\ C[j]>0}}^{255} \frac{C[j]}{w} log_2\left(\frac{C[j]}{w}\right)$
$H[0] \leftarrow H_{cur}$
$q \leftarrow 1$
for k ***in*** $[1, p_{max})$ ***step*** v
  for j ***in*** $[0, v)$
    if $S[k] \neq S[k + w]$
      $c_t \leftarrow C[S[k]]$
      $c_k \leftarrow C[S[k + w]]$
      if $c_t > 1$
        $H_{cur} \leftarrow H_{cur} + \frac{c_t \log_2(c_t) - (c_t - 1)\log_2(c_t - 1)}{n}$
      end if
      if $c_k > 0$
        $H_{cur} \leftarrow H_{cur} - \frac{(c_k + 1)\log_2(c_k + 1) - c_k \log_2(c_k)}{n}$
      end if
      $C[S[k]] \leftarrow c_t - 1$
      $C[S[k + w]] \leftarrow c_k + 1$
    endif
  end for
  $H[q] \leftarrow H_{cur}$
  $q \leftarrow q + 1$
end for

Оценим вычислительную сложность представленного алгоритма с точки зрения количества операций доступа к символам сообщения S. На первом шаге алгоритма данная операция выполняется w раз. Далее, на каждом шаге внутреннего цикла происходит 2 обращения – к добавляемому и удаляемому символам. Таким образом, всего выполняется $w + 2\big((n - w + 1) - 1\big)$ обращений. Иными словами, вычислительная сложность алгоритма равна $O(n)$.

В целях валидации разработанного алгоритма проведена его экспериментальная оценка, направленная на сравнение производительности классического и разработанного алгоритмов. В качестве исходных данных экспериментов использованы выборки размером 512 байт из файлов различного типа, входящие в массив данных FFT-75 [17]. Выборки были сгруппированы по

типу файла, после чего была произведена конкатенация каждой группы.

Первый эксперимент направлен на оценку точности разработанного алгоритма путем определения значения средней квадратической ошибки (СКО) между результатами работы оригинального и разработанного алгоритмов. Усредненная величина СКО по всем файлам набора FFT-75 с различными значениями ширины окна и смещением, равным одному символу, представлена в таблице 5. Видно, что значение СКО находится ниже уровня погрешности типа данных с плавающей запятой двойной точности, что свидетельствует в пользу высокой точности разработанного алгоритма.

Таблица 5 – Средняя квадратическая ошибка для различных значений ширины окна

w	256 байт	512 байт	1024 байт	2048 байт	4096 байт
СКО	$4.18*10^{-28}$	$5.78*10^{-27}$	$1.51*10^{-26}$	$3.69*10^{-28}$	$2.00*10^{-27}$

Второй эксперимент направлен на проверку утверждения о вычислительной сложности оригинального и разработанного алгоритмов в худшем случае для оригинального алгоритма. Параметры скользящего окна, выбранные для данного эксперимента, перечислены в таблице 6 (размер файла изменяется от 256 до 16384 байт).

Таблица 6 – Параметры скользящего окна

Параметр	w	v	n
Значение	$n/2$	128 байт	Переменная

В ходе эксперимента производился расчет динамики энтропии для файлов из FFT-75 различного размера с измерением количества операций доступа к памяти. Результаты данного эксперимента представлены на рисунке 3.

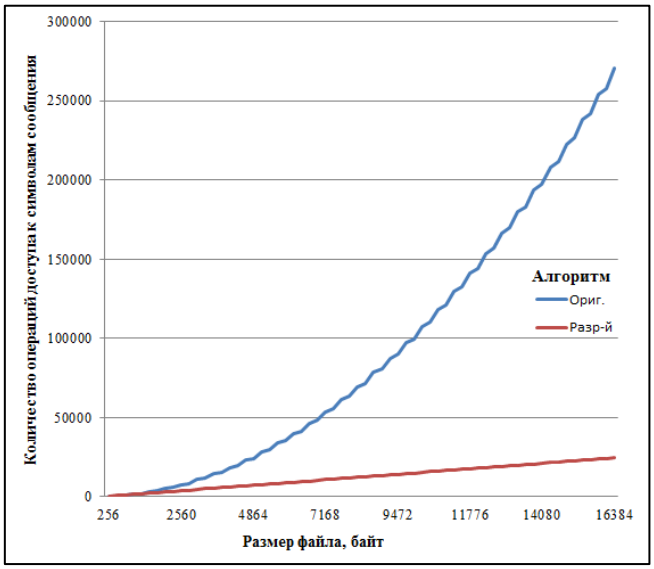

Рис. 1 – График производительности алгоритмов для худшего случая

На рисунке 3 можно заметить квадратичный тренд оригинального алгоритма и линейный тренд разработанного алгоритма, что подтверждает предположение о более высокой производительности разработанного алгоритма в худшем случае.

Третий эксперимент направлен на установление зависимости производительности оригинального и разработанного алгоритмов от величины смещения в среднем случае для оригинального алгоритма. Параметры скользящего окна, выбранные для данного эксперимента, перечислены в таблице 7 (смещение изменяется от 32 до 256 байт).

Таблица 7 – Параметры скользящего окна

Параметр	w	v	n
Значение	256 байт	Переменная	65536 байт

В ходе эксперимента производился расчет динамики энтропии для файлов из FFT-75 различного размера с измерением количества операций доступа к памяти. Результаты данного эксперимента представлены на рисунке 4.

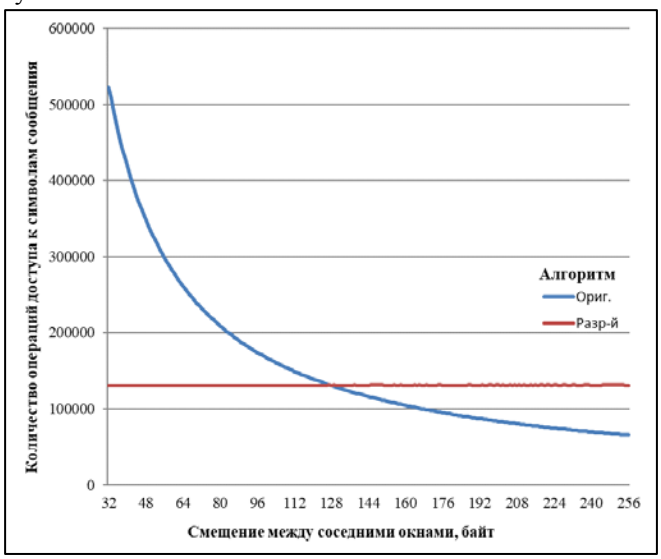

Рис. 2 – График производительности алгоритмов для среднего случая

Из рисунка 4 видно, что при увеличении величины смещения оригинальный алгоритм становится более производительным по отношению к разработанному. Более того, начиная с определенного значения смещения разработанный алгоритм уступает в производительности оригинальному.

IV. Обсуждение

С целью определения условия выбора наиболее производительного алгоритма рассмотрим соотношение количества операций доступа к символам входного сообщения, выполняемых в оригинальном (N_0) и разработанном (N_1) алгоритмах:

$$\begin{cases} N_0 = w * \left\lfloor \frac{n-w+1}{v} \right\rfloor \\ N_1 = 2n - w \end{cases}$$

Очевидно, что при $N_0 < N_1$ необходимо использовать оригинальный алгоритм, а при $N_0 \geq N_1$ – модифицированный.

Пусть $N_0 = N_1$. Тогда:

$$w * \left\lfloor \frac{n-w+1}{v} \right\rfloor = 2n - w$$

$$\left\lceil \frac{n-w+1}{v} \right\rceil = \frac{2n}{w}$$

$$\frac{2n-w}{w} \le \frac{n-w+1}{v} < \frac{2n}{w}$$

$$\frac{w(n-w+1)}{2n} < v \le \frac{w(n-w+1)}{2n-w}$$

Рассуждения при $N_0 < N_1$ и $N_0 > N_1$ аналогичны. Соотношение производительности алгоритмов в зависимости от величины смещения v проиллюстрировано рисунком 5.

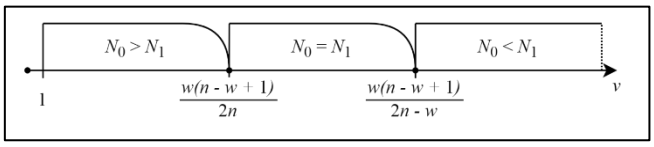

Рис. 3 – Зависимость производительности алгоритмов от величины смещения

Из рисунка 5 видно, что разработанный алгоритм является не менее производительным, чем оригинальный, при $v \le \frac{w(n-w+1)}{2n-w}$; в противном случае целесообразно использовать оригинальный алгоритм. Таким образом, в инструментах энтропийного анализа рационально использовать комбинированный алгоритм, в котором выбор способа расчета происходит в зависимости от соотношения между параметрами скользящего окна.

Идея алгоритма заключается в следующем. Пусть после некоторой итерации алгоритма скользящего окна длина необработанной части сообщения равна n. Отметим, что в ходе работы алгоритма данная величина монотонно и равномерно убывает. При условии фиксированных v и w на каждой итерации возможно определить более производительный алгоритм для использования на следующей итерации. Соотношение производительности алгоритмов в зависимости от длины необработанной части сообщения представлено на рисунке 6.

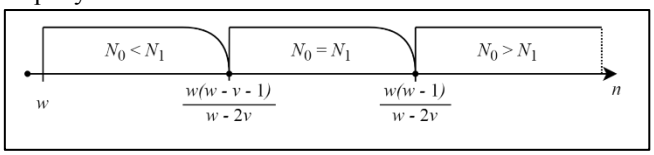

Рис. 4 – Зависимость производительности алгоритмов от длины сообщения

Из рисунка 6 следует, что при убывающей длине n в процессе выполнения алгоритма возможен единственный переход от разработанного алгоритма к оригинальному, которым будет обработан «хвост» сообщения S.

V. Заключение

Разработанный алгоритм расчета скользящей энтропии обладает низкой вычислительной сложностью, что позволяет увеличить производительность инструментов анализа энтропии двоичных файлов. Данный алгоритм также может быть использован при обработке потоковой информации (например, цифровых сигналов) в реальном времени.

Основное направление развития данного исследования состоит в обеспечении многомасштабной обработки сообщения алгоритмом – одновременного расчета энтропии с несколькими значениями ширины окна. Кроме того, алгоритм может быть улучшен путем добавления возможности адаптивной подстройки размера блока.

Библиография

[1] Cortesi A. Visualizing entropy in binary files [Электронный ресурс] // Блог Aldo Cortesi. 2012. URL: https://corte.si/posts/visualisation/entropy/ (дата обращения: 16.11.2022).

[2] Нестерович С.А., Купцова Ю.И. О некоторых возможностях обнаружения скрытого вредоносного кода // Вестник Российского нового университета. Серия: Сложные системы: модели, анализ и управление. – 2021. – № 3. – С. 156-161. – DOI 10.25586/RNU.V9187.21.03.P.156.

[3] Benvenuto F. et al. Firmware Extraction from Real IoT Devices through Power Analysis of AES //ITASEC. – 2021. – С. 461-474.

[4] Hui T.X., Mohamad K.M., Rahman N.H.A. myEntropy: a file type identification tool using entropy scoring //International Journal of Electronic Security and Digital Forensics. – 2022. – Т.14. – № 1. – С. 76-95.

[5] Копыльцов А.В. Вейвлет-анализ структурной энтропии файлов //Известия Российского государственного педагогического университета им. А.И. Герцена. – 2011. – № 138. – С. 7-15.

[6] Сорокин И.В. Математические модели и алгоритмы распознавания упакованных вредоносных программ: дис. – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, 2013.

[7] Югансон А.Н. Метод определения упакованных и зашифрованных данных во встроенном программном обеспечении //Научно-технический вестник информационных технологий, механики и оптики. – 2020. – Т. 20. – №. 5. – С. 708-713.

[8] Lyda R., Hamrock J. Using entropy analysis to find encrypted and packed malware //IEEE Security & Privacy. – 2007. – Т. 5. – № 2. – С. 40-45.

[9] Ugarte-Pedrero X. et al. Countering entropy measure attacks on packed software detection //2012 IEEE Consumer Communications and Networking Conference (CCNC). – IEEE, 2012. – С. 164-168.

[10] Choi M.J. et al. All-in-one framework for detection, unpacking and verification for malware analysis //Security and Communication Networks. – 2019. – Т. 2019. – DOI: 10.1155/2019/5278137.

[11] BinGraph [Электронный ресурс] // Github. URL: https://github.com/geekscrapy/binGraph (дата обращения: 16.11.2022).

[12] Bintropy [Электронный ресурс] // Github. URL: https://github.com/packing-box/binropy (дата обращения: 16.11.2022).

[13] Binwalk [Электронный ресурс] // Github. URL: https://github.com/ReFirmLabs/binwalk (дата обращения: 16.11.2022).

[14] Detect-It-Easy [Электронный ресурс] // Github. URL: https://github.com/horsicq/Detect-It-Easy (дата обращения: 16.11.2022).

[15] EntroPy [Электронный ресурс] // Github. URL: https://github.com/gcmartinelli/entroPy (дата обращения: 16.11.2022).

[16] Ghidra [Электронный ресурс] // Github. URL: https://github.com/NationalSecurityAgency/ghidra (дата обращения: 16.11.2022).

[17] Mittal G., Korus P., Memon N. FiFTy: large-scale file fragment type identification using convolutional neural networks //IEEE Transactions on Information Forensics and Security. – 2020. – Т. 16. – С. 28-41.

Methodology of software expertise based on improved windowed entropy calculation algorithm

S.V. Karbovskiy

***Abstract*— The article discusses an algorithm for calculating the sliding entropy of a binary file with an intersection between adjacent blocks. The relevance of the study is due to the widespread use of entropy analysis in software expertise. The currently used algorithm for calculating entropy with intersection between adjacent blocks has quadratic complexity in the worst case. As a result, the entropy calculation algorithm used in the analysis tools divides the input file into disjoint blocks, which reduces the accuracy of entropy analysis. The paper analyzes the changes in the information entropy of a message when one character changes in it. It is determined that calculations of entropy changes, regardless of the ratio of the frequency of occurrence of deleted and added characters in the message, can be carried out in constant time. As a result of the analysis of the revealed dependencies, a more productive algorithm for calculating the sliding entropy of binary files with intersecting blocks has been developed. It is shown that the developed algorithm makes it possible to calculate entropy with an arbitrary amount of displacement between adjacent blocks of the file. An experimental evaluation of the accuracy and performance of the algorithm was carried out. It is revealed that the performance gain when using the developed algorithm increases with a decrease in the offset between adjacent blocks. The study is intended for specialists in the field of system analysis of software, as well as in the field of reverse engineering of software.**

***Keywords*—entropy analysis, moving window entropy, algorithm analysis, reverse engineering.**

REFERENCES

[1] Cortesi A. Visualizing entropy in binary files // Aldo Cortesi. 2012. URL: https://corte.si/posts/visualisation/entropy/ (visited: 16.11.2022).

[2] Nesterovich S.A., Kuptsova Yu.I. On some possibilities of hidden malicious code detection //Proceedings of the Russian New University. Serie: Complex Systems: models, analysis and control. – 2021. – № 3. – C. 156-161. – DOI 10.25586/RNU.V9187.21.03.P.156.

[3] Benvenuto F. et al. Firmware Extraction from Real IoT Devices through Power Analysis of AES //ITASEC. – 2021. – C. 461-474.

[4] Hui T.X., Mohamad K.M., Rahman N.H.A. myEntropy: a file type identification tool using entropy scoring //International Journal of Electronic Security and Digital Forensics. – 2022. – T.14. – № 1. – C. 76-95.

[5] Kopyltsov A.V. Wavelet-analysis of the file structural entropy //Proceedings of Russian State Pedagogical University. – 2011. – № 138. – C. 7-15.

[6] Sorokin I.V. Mathematical models and algorithms of packed malicious program recognition: diss. – Saint-Petersburg National Research University of Information Technologies, Mechanics and Optics, 2013.

[7] Yuganson A.N. Method for determining packed and encrypted data inside of embedded software //Scientific and Technical Bulletin of Information Technologies, Mechanics and Optics. – 2020. – T. 20. – №. 5. – C. 708-713.

[8] Lyda R., Hamrock J. Using entropy analysis to find encrypted and packed malware //IEEE Security & Privacy. – 2007. – T. 5. – № 2. – C. 40-45.

[9] Ugarte-Pedrero X. et al. Countering entropy measure attacks on packed software detection //2012 IEEE Consumer Communications and Networking Conference (CCNC). – IEEE, 2012. – C. 164-168.

[10] Choi M.J. et al. All-in-one framework for detection, unpacking and verification for malware analysis //Security and Communication Networks. – 2019. – T. 2019. – DOI: 10.1155/2019/5278137.

[11] BinGraph // Github. URL: https://github.com/geekscrapy/binGraph (visited: 16.11.2022).

[12] Bintropy // Github. URL: https://github.com/packing-box/binropy (visited: 16.11.2022).

[13] Binwalk // Github. URL: https://github.com/ReFirmLabs/binwalk (visited: 16.11.2022).

[14] Detect-It-Easy // Github. URL: https://github.com/horsicq/Detect-It-Easy (visited: 16.11.2022).

[15] EntroPy // Github. URL: https://github.com/gcmartinelli/entroPy (visited: 16.11.2022).

[16] Ghidra // Github. URL: https://github.com/NSA/ghidra (visited: 16.11.2022).

[17] Mittal G., Korus P., Memon N. FiFTy: large-scale file fragment type identification using convolutional neural networks //IEEE Transactions on Information Forensics and Security. – 2020. – T. 16. – C. 28-41.