Исследование метода аутентификации двоичных изображений при помощи технологии ЦВЗ

Д.т.н., проф. Коржик В.И.
Студентка группы ИКТЗ-23
Тришневская И.А.

Идея метода

- Разделяем двоичное изображение (ДИ) на блоки 3x3 пикселя $X = (x_{ij})_{1\le i,\, j\le 3}$, где $x_{ij} = 1$, если ij-пиксель черный и $x_{ij} = 0$, если ij-пиксель белый
- Определяем дискриминационное отображениеd: $X \longrightarrow d(X)$

$$d(X) = \sum_{i=1}^{3}\sum_{j=1}^{2}|x_{i,j+1} - x_{ij}| + \sum_{j=1}^{3}\sum_{i=1}^{2}|x_{i+1,j} - x_{ij}|$$

- Определяем обратную операцию – *flipping* – инвертирование центрального пикселя

  $f: X \longrightarrow f(X) = Y = (y_{ij})_{1\le i,\, j\le 3}$

$$y_{ij} = \begin{cases} x_{ij} & \text{если } (i,j) \neq (2,2) \\ |\,1 - x_{ij}\,| & \text{если } (i,j) = (2,2) \end{cases}$$

- Определяем 3 типа областей:

  Регулярные $R = \{X \mid d(X) \neq 0 \;\&\; d(f(X)) > d(X)\}$

  Сингулярные $S = \{X \mid d(f(X)) \neq 0 \;\&\; d(f(X)) < d(X)\}$

  Неиспользуемые $U = \{X \mid d(X) = 0 \text{ or } d(f(X)) = 0 \text{ or } d(f(X)) = d(X)\}$

- Формируем RSU-последовательность σ и назначаем R-область 1, S-область 0, U-область – nil
- Удаляем из последовательности nil-области
- Сжимаем {0,1}-последовательность (используем сжатие без потерь – адаптированное арифметическое кодирование)
- Формируем код идентификации сообщения (message authentication code – MAC) – последовательность двоичных символов, полученных из ДИ криптографическим алгоритмом (длинна 64-256 бит)
- Проверяем, можно ли добавить MAC к сжатой последовательности

  Если $n_T - n_C \geq n_0$, дописываем после сжатой последовательности MAC

  n_T – общее число R и S областей, n_C – длина сжатой последовательности, n_0 – длина MAC
- Трансформируем {0,1}-последовательность в RS. (U-области сохраняются неизменными)
- Если элементы полученной последовательности и исходной не совпадают, меняем значение центрального пикселя в соответствующих областях.

Проверка подлинности

- Дано ДИ с вложенным MAC. Формируем RSU-последовательность и назначаем 1, 0 и nil, как при вложении
- {0,1}-битный поток разделяем на MAC и сжатый {R,S}-вектор. Декодируем последовательность
- Изображение обрабатывается с целью регулирования состояния всех R и S областей, инвертируя, если это необходимо, центральный пиксель в исходное положение. Таким образом мы получаем точную копию исходного сообщения
- Формируем MAC, соответствующий полученному изображению и сравниваем с извлеченным в п.2

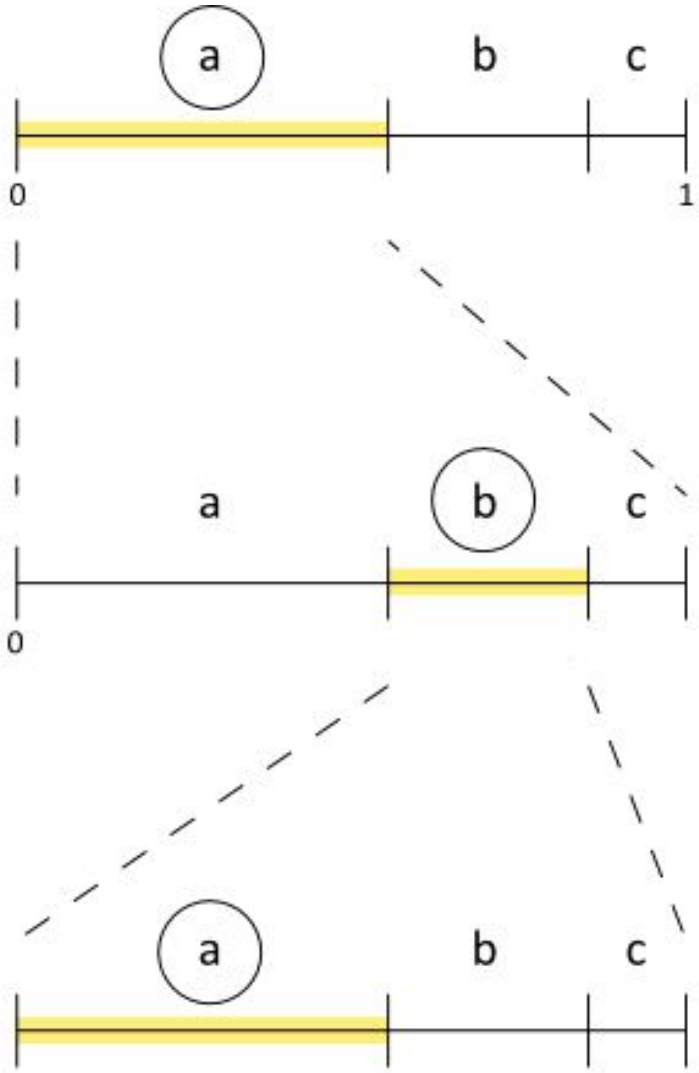
  Если MAC совпадают, значит ДИ подлинное

Арифметическое кодирование

- Арифметическое кодирование (Arithmetic coding) — алгоритм сжатия информации без потерь, который при кодировании ставит в соответствие тексту вещественное число из отрезка [0;1). Данный метод (как и алгоритм Хаффмана) является энтропийным т.е. длина кода конкретного символа зависит от частоты встречаемости этого символа в тексте. Арифметическое кодирование показывает более высокие результаты сжатия, чем алгоритм Хаффмана, для данных с неравномерными распределениями вероятностей кодируемых символов.
- При кодировании алгоритму передается только текст для кодирования
- При декодировании алгоритму передается закодированный текст и длина исходного текста

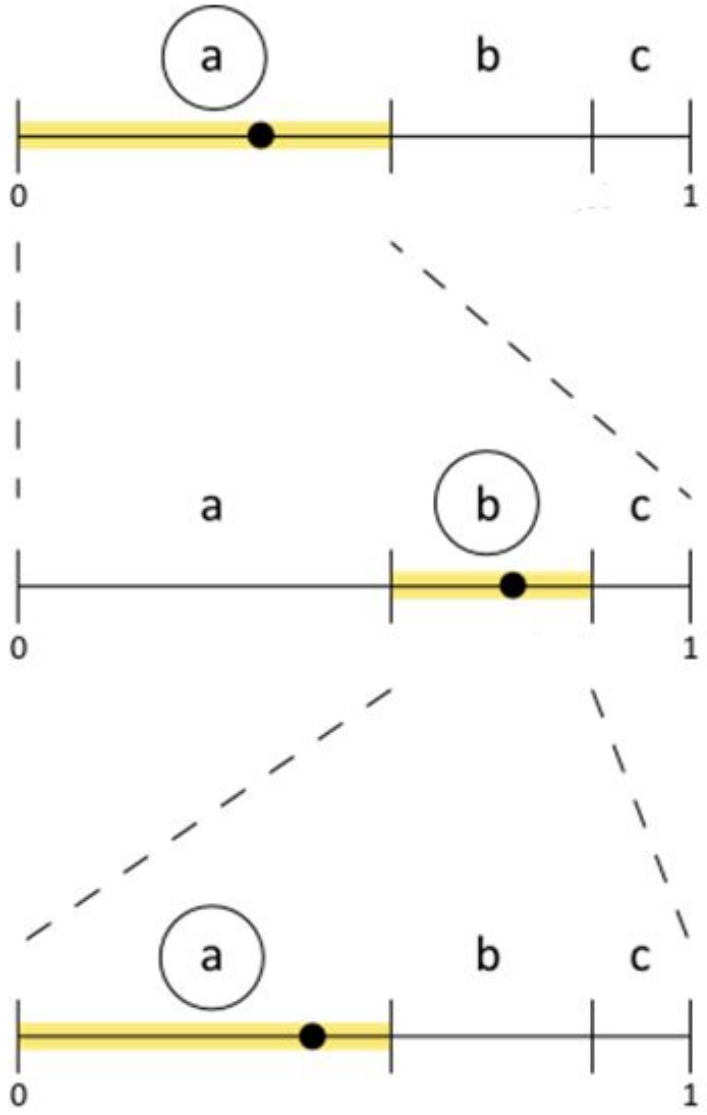
Кодирование

- Рассмотрим отрезок [0,1) на координатной прямой
- Поставим каждому символу текста в соответствие отрезок, длина которого равна частоте его появления.
- Считаем символ из входного потока и рассмотрим отрезок, соответствующий этому символу. Этот отрезок разделим на части, пропорциональные частотам встречаемости символов.
- Повторим пункт (3) до конца входного потока.
- Выберем любое число из получившегося отрезка. Это и будет результат арифметического кодирования.



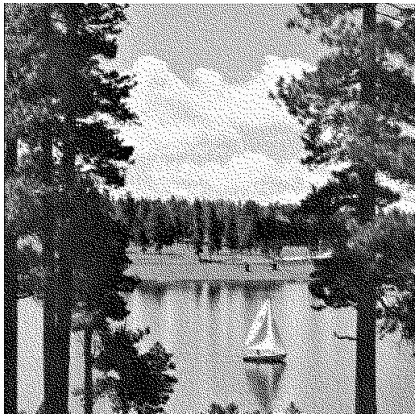
Декодирование

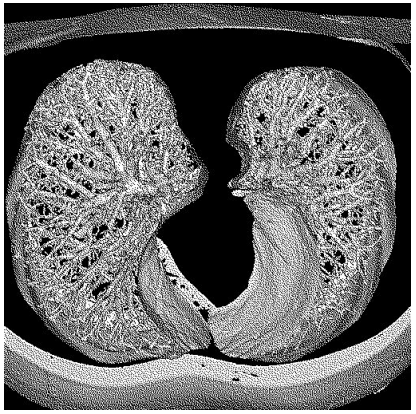
- Выберем на отрезке [0, 1), разделенном на части, длины которых равны вероятностям появления символов в тексте, подотрезок, содержащий входное вещественное число. Символ, соответствующий этому подотрезку, дописываем в ответ.
- Нормируем подотрезок и вещественное число.
- Повторим п. (1-2) до тех пор, пока не получим ответ (до конца файла).



Результаты исследования

Images	N_R	N_S	N_U	N_T	length of encoded sequence	free
0	10465	5367	13068	15832	14635	1197
1	9186	9246	10468	18432	18440	-8
2	15193	4557	9150	19750	15399	4351
3	6642	11571	10687	18213	17246	967
4	8641	9036	11223	17677	17678	-1
5	13191	5208	10501	18399	15823	2576
6	10537	7512	10851	18049	17690	359
7	11198	7748	9954	18946	18499	447
8	3343	4942	20615	8285	8069	216
9	15206	1687	12007	16893	7924	8969
10	16864	1396	10640	18260	7122	11138





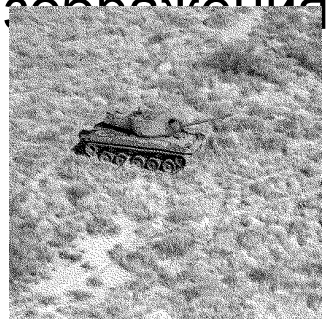
N_R,N_S N_U – количество R,S U областей соответственно

$N_T = N_R + N_S$

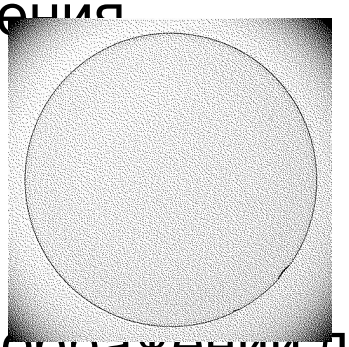

Количество изображений
25
20
15
10
5
0
< 0
0-64
64-500
501-1000
1001-1500
1501-2000
2001-2500
2501-3000
3001-3500
3501-4000
4001-4500
4501-5000
5001-5500
5501-6000
6001-6500
6501-7000
7001-7500
7501-8000
8001-8500
8501-9000
9001-10000
10001-11000
11001-12000
12001-13000
Количество свободных бит

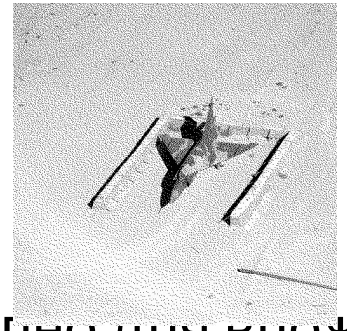
Выводы по таблице

- АК дает нам достаточно места для аутентификатора
- Меньше всего подходят для вложений пёстрые и зашумленные изображения



- Больше места для вложения дают более однотонные и четкие изображения





- Чаще всего в изображении доступно для вложения 64-500 бит (стандартный аутентификатор 64-128 бит)
- Около 18% изображений из исследуемой базы непригодны для вложения стандартного аутентификатора

Текстурность изображения

$$t_n = \frac{1}{n_1 \times n_2} \sum_{i,j} \left(\max_k B_{ij}^k - \min_k B_{ij}^k \right)$$

n1, n2 – размер изображения

B_{ij}^k - область 2x2 пикселя, где ij – координаты области, k – k-ый пиксель области

Images	free	image texturing	Images	free	image texturing
26	93	0.003	25	12854	0.097
28	89	0.003	8	216	0.1
14	1422	0.005	59	536	0.102
15	1894	0.054	66	8267	0.103
12	8247	0.059	38	15	0.105
35	3805	0.067	9	8969	0.108
46	1151	0.068	10	11138	0.11
36	1974	0.075	19	570	0.118
93	5519	0.091	98	4812	0.119
11	286	0.093	22	2508	0.121

Вывод: связи между текстурностью изображения и количеством освобождаемых бит не выявлено

Итоги

- Изучено арифметическое кодирование. Реализован адаптивный кодер на Java
- Произведен расчет количества R, S и U областей, количество освобождаемых после кодирования бит и значения текстурности для 100 изображений
- Написан код на Java для автоматизации данного процесса
- Исследована связь между значением текстурности и количеством свободных для вложения бит. Связи не установлено

Перспективы

- Реализация добавления аутентификатора к двоичному изображению
- Улучшение метода
- Установление связи между параметрами изображения и его пригодностью для вложения аутентификатора

Список литературы

1. Valery Korzhik, Guillermo Morales-Luna and Michail Zubarev “Distortion Free Exact Authentication of Binary Images” journal of latex class files, vol.6, no. 1, January 2007
2. Valery Korzhik, Guillermo Morales-Luna, Ivan Fedyanin “Using Generalized Viterbi Algorithm to Perform Highly Effective Stegosystem for Images”
3. Б.Д.Кудряшов учебник для вузов «Теория информации» 2009г