Санкт–Петербургский государственный университет

Блинов Иван Сергеевич

Выпускная квалификационная работа

Применение алгоритмов разделения секрета к кодированию элементов веб-контента

Уровень образования: бакалавриат
Направление 01.03.02 «Прикладная математика и информатика»
Основная образовательная программа СВ.5005.2018 «Прикладная математика, фундаментальная информатика и программирование»
Профиль «Исследование и проектирование систем управления и обработки сигналов»

Научный руководитель:
профессор, кафедра управления медико-биологическими системами, д.ф. - м.н. Утешев Алексей Юрьевич

Рецензент:
профессор, кафедра компьютерного моделирования и многопроцессорных систем, д.т.н. Дегтярев Александр Борисович

Санкт-Петербург
2022 г.

Содержание

Введение

В настоящее время JavaScript является одним из самых популярных языков программирования. Он широко используется для создания приложений, исполняемых и со стороны клиента в браузере, и со стороны сервера. Так же распространены нативные приложения для мобильных устройств, Progressive Web Apps - гибриды нативных приложений и сайтов. В первую очередь это связано с развитием интернета и увеличением объема передаваемых по сети данных. Вместе с этим растет потребность в безопасности данных, которые представляют собой некоторую ценность. Потребность передачи секретных данных возникает у ученых, военных, в судопроизводстве, бизнесе. Традиционные методы защиты информации предоставляет криптография. Чаще всего информация защищается с помощью секретного алгоритма или ключа. Но у такого подхода есть проблемы: если злоумышленник перехватит ключ или скомпрометирует одну из сторон, то он легко получит доступ к секрету. Также, при необходимости разделить секретную информацию между какой-то группой людей приходится устанавливать соединения между каждой парой из группы, что негативно сказывается на безопасности секрета.

В 1979 году A. Shamir представил [1] алгоритм разделения секрета, который позволяет разбить секрет на n долей таким образом, что знание K и более долей позволяет восстановить секрет, а знание $K - 1$ и менее долей делает восстановление секрета невозможным. В последние десятилетия было предложено множество алгоритмов разделения секрета для электронных изображений. В данной работе будет рассмотрен алгоритм обратимого разделения секрета для изображения в оттенках серого и адаптирован для использования с цветными секретными изображениями, реализована библиотека для использования в веб-приложениях и пример минимального проекта, использующего эту библиотеку.

Исходя из вышесказанного, актуальность работы заключается в необходимости разработки программных средств с открытым исходным кодом для обеспечения безопасности веб-приложений с возможностью разделения секретного изображения на несколько частей. Открытый исходный код позволяет пользователям убедиться в отсутствии вредоносных скриптов, уязвимостей,

дает возможность искать ошибки и дополнять код сообществу разработчиков.

Цель и постановка задачи

Целью данной работы является написание библиотеки для языка JavaScript, для разделения секретного цветного электронного изображения, с долями, не подобными шуму. Для достижения этой цели были поставлены следующие задачи:

1. Исследование предметной области;
2. Выбор алгоритма;
3. Модификация алгоритма для работы с цветным секретным изображением;
4. Написание библиотеки;
5. Написание минимального веб-приложения, позволяющего продемонстрировать работу программы;
6. Тестирование библиотеки.

Обзор литературы

Основной работой является схема разделения секрета Шамира [1], более подробно рассмотренная в следующей главе.

Статья «An investigation on image secret sharing »[2] структурирует разновидности модификаций схемы Шамира и содержит многие выдержки из современных работ.

Основным алгоритмом в этой работе является Reversible Image Secret Sharing [3] из одноименной статьи.

При работе использовался ресурс Scopus [4] для поиска научных статей. Для имплементации библиотеки и веб-приложения использовались open-source библиотеки и документации [5] [6] [7] к ним.

Глава 1. Исследование предметной области

Одним из первых алгоритмов разделения секрета является (k, n) пороговая схема Шамира(ссылка). В ее основе лежит интерполяция полиномов. Пусть D – некоторая секретная информация, представленная в форме числа. Выберем простое число $p : p > D, p > N$. Чтобы разделить секрет на n частей возьмем случайный полином степени $k - 1$

$$q(x) = a_0 + a_1 x + ... + a_{k-1} x^{k-1}, a_0 = D, a_i < p$$

и вычислим

$$D_1 = q(1) \bmod p, ..., D_i = q(i) \bmod p, ..., D_n = q(n) \bmod p$$

Число p будет публичным для всех участников, числа D_i назовем долями. Участника схемы, который хочет разделить секрет и формирует доли назовем дилером.

Имея k и более долей, можно восстановить секрет D при помощи полиномиальной интерполяции.

Допустим, злоумышленнику удалось получить доступ к $k - 1$ долям, тогда для каждого $D' : 0 < D' < p$ он может восстановить единственный полином степени $k-1$, такой, что $q_0 = D'$ и $q_i = D_i$. Так как a_i случайны, эти p полиномов с одинаковой вероятностью являются искомыми, злоумышленник не получает никакой информации о секрете.

Схема Шамира позволяет разделить секрет, представленный в форме числа и используется в основном для защиты ключей. Изображение так же можно представить в форме числа, но при обычном размере изображения (для примера 256x256) и значениях пикселя (0-255)x3 для RGB изображений будет тратиться огромное количество памяти и возрастет время генерации долей и восстановления секрета. Поэтому на основе схемы Шамира были разработаны алгоритмы разделения секрета для изображений. Их можно разделить на три категории - схемы визуальной криптографии(VCS), полиномиальные схемы и схемы, основанные на Китайской теореме об остатках.

В 1994 году Moni Naor и Adi Shamir (ссылка) представили первую VCS,

на ее основе были разработаны другие модификации. В VCS схемах доли обычно печатаются на прозрачных носителях и секрет восстанавливается путем наложения частей друг на друга. Основным преимуществом таких схем является отсутствие необходимости вычислений при восстановлении секрета. Примечательной для применения в веб-разработке является VCS схема WEB-VC (ссылка). Алгоритм восстановления секрета в ней основан на возможности установить прозрачность элемента в таблице каскадных стилей (CSS). Основными недостатками таких схем являются наличие помех в восстановленном секретном изображении и возможность использования только с бинарными изображениями.

Полиномиальные схемы используются чаще из-за лучшего качества восстановленного секрета и в общем случае не требуют увеличения количества пикселей. Но у них есть и недостатки – относительно высокая вычислительная сложность восстановления секрета $O(k \times log^2(k))$ для каждого пикселя и небольшие потери в качестве восстановленного секретного изображения.

Схемы, основанные на Китайской теореме об остатках имеют более низкую сложность операции восстановления $O(k)$ и позволяют восстановить секрет без потерь. Недостатком таких схем является ограниченное количество участников(ссылка). Таким образом, эти схемы отлично подходят для устройств с низкой вычислительной мощностью и для испозьзования в веб-приложениях.

Во многих схемах дилер отправляет участникам шумо-подобные доли. Введём понятие изображений для прикрытия – это произвольные изображения, использующиеся для генерации долей. Сгенерированные доли являются изображениями, похожими на изображения прикрытия. Использование изображений прикрытия вместо шумо-подобных долей снижает риск привлечения внимания к долям злоумышленников, улучшает возможности по их менеджменту.

В данной работе будет рассматриваться алгоритм Reversible Image Secret Sharing [3]. Он основан на китайской теореме об остатках. В качестве секретной картинки выступает изображение в оттенках серого $(0-255)$, картинками для прикрытия являются бинарные изображения.

Глава 2. Описание алгоритма

Начнем описание работы алгоритма с формулировки Китайской теоремы об остатках.

Если $a_1, ..., a_n \in N$ попарно взаимно просты, то для

$$\forall r_1, ..., r_n \in N : 0 \leq r_i < a_i, \forall i \in \overline{1, n}$$

найдется $N : N \bmod a_i = p_i, \forall i \in \overline{1, n}$

Эта теорема позволяет за линейное время решать систему линейных модулярных уравнений следующего типа:

$$\begin{cases} y \equiv a_1 \bmod m_1 \\ y \equiv a_2 \bmod m_2 \\ ... \\ y \equiv a_k \bmod m_k \end{cases}$$

Алгоритм решения (ссылка):

1. Вычисляем $M = \prod_{i=1}^{k} m_i$;
2. $\forall i \in \overline{1, k}$ вычисляем $M_i = \frac{M}{m_i}$;
3. С помощью расширенного алгоритма Евклида $\forall i \in \overline{1, k}$ находим ${M_i}^{-1}$ обратное по модулю для M_i;
4. Получаем $y \equiv \sum_{i=1}^{k} a_i M_i {M_i}^{-1} \bmod M$.

Предложенный алгоритм состоит из двух частей: формирование долей и восстановление секрета. Опишем их более подробно.

2.1 Формирование долей

Описание входных данных:

• Секретное изображение S размера $W_S \times H_S$ пикселей в оттенках серого (значения пикселей 0-255);

• n - количество долей;

• k - минимальное количество долей для восстановления секрета;

• n изображений C_i размера $W_S \times H_S$ - бинарные (значения пикселей 0-1) изображения прикрытия для каждого из участников.

Описание выходных данных:

• n изображений SC_i размера $W_S \times H_S$ - сгенерированные доли;

• m_i - приватное число для каждой доли;

• p, T - публичные для всех участников числа для восстановления секрета.

Алгоритм:

1. Выберем число p и n взаимно простых чисел m_i таких, что

$$128 \leq p < m_i \leq 256, \text{НОД}(m_i, p) = 1, \forall i \in \overline{1, n}$$

2. Вычислим $M = \prod_{i=1}^{k} m_i$, $N = \prod_{i=1}^{k-1} m_{n-i+1}$

3. Если $M < pN$ перейдем к шагу 1

4. Вычислим $T = \left[\frac{\left\lfloor \frac{M}{p} - 1 \right\rfloor}{2} \right]$

5. Для каждого секретного пикселя x с координатами $[w, h]$ повторяем шаги 6-7

6. Если $0 \leq x < p$, выберем случайное $A \in \left[T + 1, \left\lfloor \frac{M}{p} - 1 \right\rfloor \right]$ и вычислим $y = x + Ap$.

Если $x \geq p$ выберем случайное $A \in [0, T)$ и вычислим $y = x - p + Ap$

7. Если выполняется одно из следующих условий, то вычисляем $a_i = y \bmod p$, устанавливаем $SC_i = a_i$ и переходим к следующему пикселю, иначе возвращаемся на шаг 6.

$$\begin{cases} SC_i[w,h] \geq TH_{i1}, \text{ если } C_i[w,h] = 1 \\ SC_i[w,h] \leq TH_{i0}, \text{ если } C_i[w,h] = 0 \end{cases}$$

$$\text{при } TH_{i0} = \frac{m_i}{2} - TH,\ TH_{i1} = \frac{m_i}{2} + TH$$

2.2 Восстановление секрета

Описание входных данных:

- n долей SC_i ($n \geq k$) и соответствующие им m_i
- Публичные числа T, p

Описаные выходных данных:

- Восстановленный секрет S'
- Восстановленные изображения прикрытия C'_i размера $[W_S, H_S]$

Алгоритм

1. Восстановливаем изображения прикрытия с помощью бинаризации. Для каждого пикселя $C'_i[w,h]$ устанавливаем значение

$$\text{Если } SC_i[w,h] > \frac{m_i}{2}, \text{ то } 1, \text{ иначе } 0$$

2. Для каждой позиции пикселя $[w,h]$, $a_i = SC_i[w,h]$, с помощью описанного выше алгоритма (ссылка) решаем систему линейных урав-

нений по модулю:

$$\begin{cases} y \equiv a_1 \bmod m_1 \\ y \equiv a_2 \bmod m_2 \\ ... \\ y \equiv a_n \bmod m_n \end{cases}$$

3. Вычисляем $T^* = \left\lfloor \frac{y}{p} \right\rfloor$. Если $T^* \leq T$, то $x = y \bmod p$, иначе $x = (y \bmod p) + p$.

$S'[w, h] = x$

2.3 Комментарии к алгоритму

1. Число p в алгоритме 1 (ссылка) выбирается наименьшим из возможных, а числа m_i выбираются наибольшими для достижения большего диапазона распределения значения пикселя в долях.

2. Параметр TH имеет весомую роль в качестве сгенерированных долей, времени генерации и безопасности. Этот параметр устанавливается дилером и влияет на N_A – число возможных значений A в шаге 6 алгоритма 1. В общем случае, $N_A = T$. Для того, чтобы удовлетворять условию на шаге 7 алгоритма, значение N_A уменьшается до $N_A = T \times \prod_{i=1}^{n} \left(\frac{1}{2} \times \frac{TH_{i0}}{m_i} + \frac{1}{2} \times \frac{m_i - TH_{i0}}{m_i} \right)$.

   При увеличении TH уменьшается N_A, увеличивается качество сгенерированных долей и время на генерацию. При увеличении N_A улучшается безопасность, так как количество значений для перебора равняется T^{N_A}. Требуется, чтобы $N_A \geq 2$, так как при $N_A = 1$ в шаге 6 алгоритма будет повторно использоваться одно и то же значение A, что приводит к проблемам с безопасностью. Экстремальной точкой для качества долей является $TH = 112$. Экспериментальные данные можно увидеть на ((Рисунке 1)).

3. Качество картинки по сравнению с изначальной будем измерять с помо-

щью пикового отношения сигнала к шуму – $PSNR$. Эту метрику чаще всего определяют с помощью среднеквадратичной ошибки MSE (ссылка). Пусть I – исходное изображение размера $m \times n$, K – зашумленная версия I. Тогда

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |I[i,j] - K[i,j]|^2$$

$$PSNR = 10 \log_{10} \left(\frac{MAX_i^2}{MSE} \right)$$

4. Мощностью встраивания EC (embedding capacity) называется отношение количества бит информации, встраиваемых в изображение, к размеру изображения и определяется формулой:

$$EC = \frac{N}{L \times H \times W}$$

N – число бит секрета, L – количество бит в пикселе, $H \times W$ – размер изображения.

Для данного алгоритма EC примет следующий вид:

$$EC = \frac{(L_S \times W_S \times H_S) + (n_C \times L_C \times W_C \times H_C)}{n_{SC} \times L_{SC} \times W_{SC} \times H_{SC}} \quad (1)$$

где S – секретное изображение, C – изображения прикрытия, SC – доли, L_x - количество бит в пикселе, $W_x,\ H_x$ – ширина и высота, n_x – количество изображений.

Таким образом, $L_S = L_{SC} = 8$, $L_C = 1$, $W_S = W_C = W_SC = W$, $H_S = H_C = H_SC = H$, $n_C = n_{SC} = n$

$$EC = \frac{(8 \times W \times H) + (n \times W \times H)}{8 \times n \times W \times H} = \frac{8+n}{8n}$$

5. Результаты работы алгоритма для изображений, размером 512×512 $n = 5$, $k = 4$, $TH = 16$ показаны на Рисунке 1. Значения $PSNR$ – SC_1: 37.36, SC_2: 37.15, SC_3: 36.93, SC_4: 37.25, SC_5: 36.98

Глава 3. Модификация алгоритма

Описанный выше алгоритм отлично подходит для цели работы, за исключением цвета секретной картинки. Поэтому было принято решение расширить исходный алгоритм для использования с цветными секретными картинками. Это было достигнуто с помощью увеличения количества пикселей в картинках прикрытия и кодирования каждого канала цвета в определенном пикселе доли. Схема расположения каналов в доле представлена на Рисунке 2.

Каждый пиксель секретного изображения кодируется в 4 пикселях картинки прикрытия. Размер исходной цветной картинки – 2×1, размер доли и размер картинки прикрытия – 8×1. Белый сегмент при использовании с картинками формата rgba отвечает за кодирование канала прозрачности alpha, для rgb не принимает участия в разделении секрета и в зависимости от значения пикселя картинки прикрытия $C_{ij} = 0|1$ принимает значения $0|255$.

Изображения в коде веб-приложения и при передаче между клиентом и сервером хранятся в форме массива бит. Для того, чтобы избежать существенного увеличения размера изображения на диске, доли можно сохранять в формате PGM – portable gray map. В нем для кодирования каждого пикселя используется 8 бит. Таким образом, изображение доли будет занимать на диске и в памяти такое же пространство, как и секретное rgba изображение.

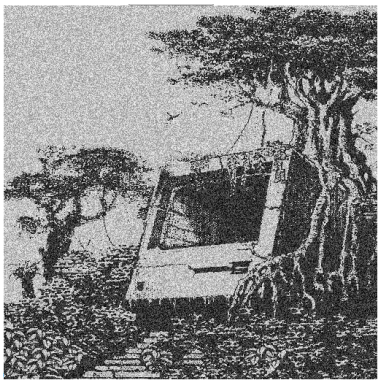
Рассмотрим метрику EC (1) для дополненного алгоритма. Для RGBA изображения:

$$EC = \frac{(32 \times \frac{W}{2} \times \frac{H}{2}) + (n \times W \times H)}{n \times 8 \times W \times H} = \frac{8+n}{8n}$$

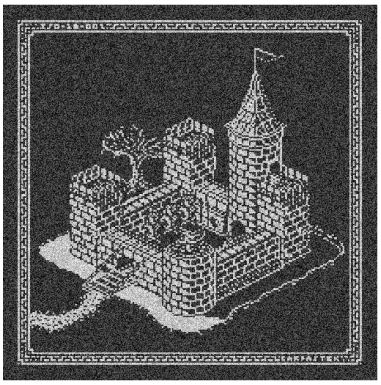
Для RGB изображения: $EC = \frac{6+n}{8n}$

Результаты работы алгоритма с цветным секретным изображением размера 256×256, $n = 4$, $k = 3$

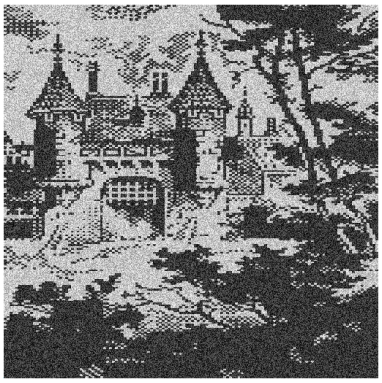
S



SC_1

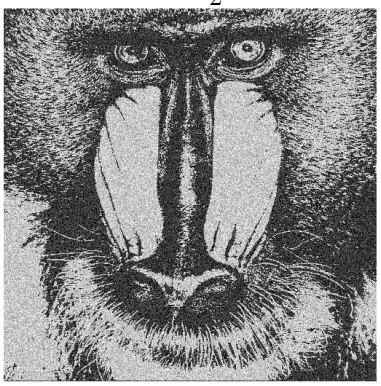


SC_2

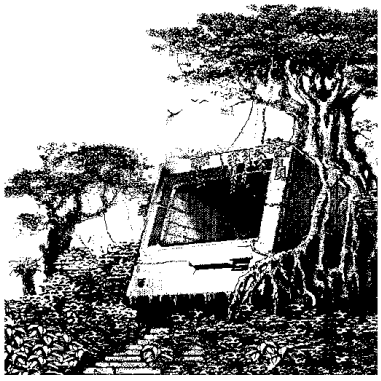


SC_3

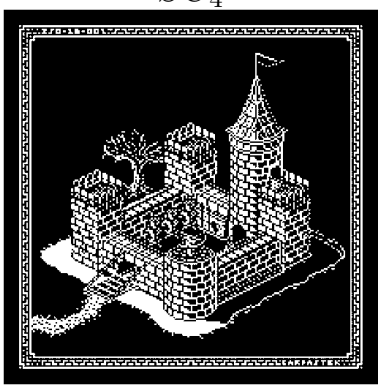
SC_4



SC_5



$C'_1 = C_1$
$PSNR = \infty$



$C'_2 = C_2$
$PSNR = \infty$

$C'_3 = C_3$
$PSNR = \infty$

$C'_4 = C_4$
$PSNR = \infty$

$C'_5 = C_5$
$PSNR = \infty$

$S'_{1234} = S'_{2345} = S$
$PSNR = \infty$

Рис. 1: Экспериментальные изображения

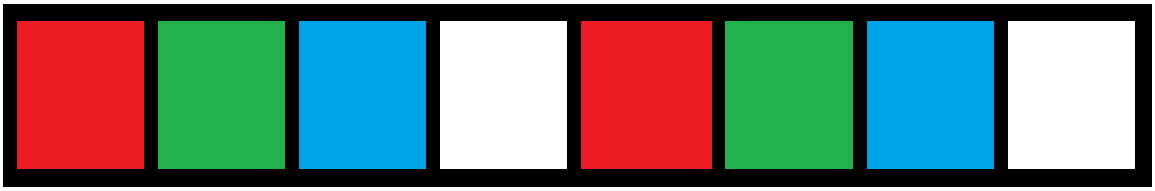


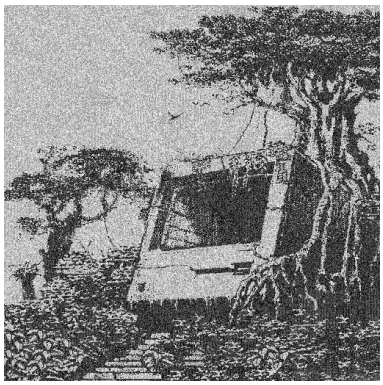
Рис. 2: Схема расположения каналов

Глава 4. Имплементация библиотеки

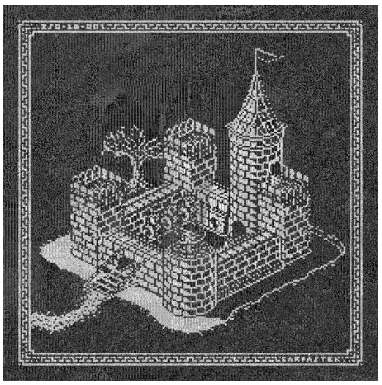
В ходе работы была написана библиотека на языке JavaScript, которая реализует работу алгоритма (ссылка 1) и адаптирует его для работы с цветными изображениями. На данный момент в открытом доступе есть две библиотеки, реализующие схему Шамира для числа: «shamir-secret-sharing» и «secrets.js», но, как было отмечено выше(ссылка?), они подходят только для кодирования секрета в форме числа. Для изображений имплементации на языке JavaScript каких-либо алгоритмов разделения секрета отсутствуют. Основным преимуществом данной библиотеки является возможность ее использования как со стороны клиента, так и со стороны сервера.

Для улучшения безопасности, все данные следуют передавать по HTTPS (HyperText Transfer Protocol Secure) - протоколу зашифрованной передачи данных. Если важна скорость работы, например для передачи большого количества изображений, то генерацию долей лучше производить на сервере. Если важнее стоит безопасность, то генерацию можно провести в клиентском приложении и отправлять доли по более защищенным каналам. Исключение сервера из процесса генерации снижает возможность атаки MITM(man in the middle). Таким образом, у злоумышленника есть только 3 варианта получения доступа к секрету - получение доступа к машине дилера, получение доступа к K частям секрета и полный перевбор значений. Последние 2 варианта являются тяжело реализуемыми на практике, поэтому основной уязвимостью системы является дилер.

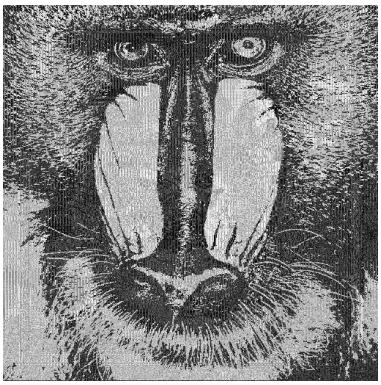
Библиотека написана в функциональном стиле, все функции не обладают побочными эффектами. Установка осуществляется с помощью пакетного менеджера npm.



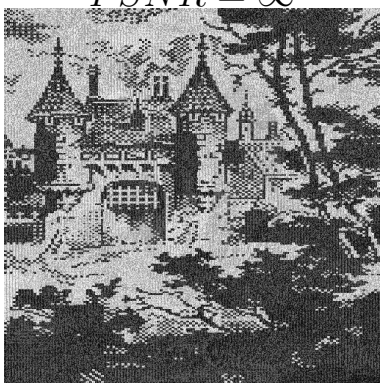
SC_1
$PSNR = \infty$



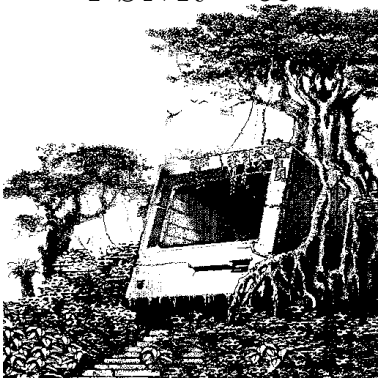
SC_2
$PSNR = \infty$



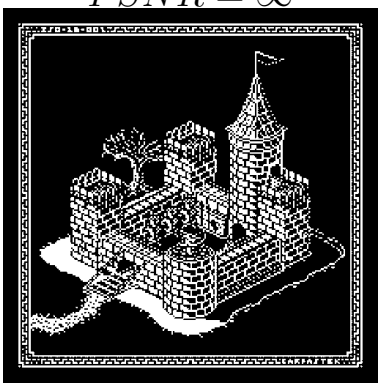
SC_3
$PSNR = \infty$



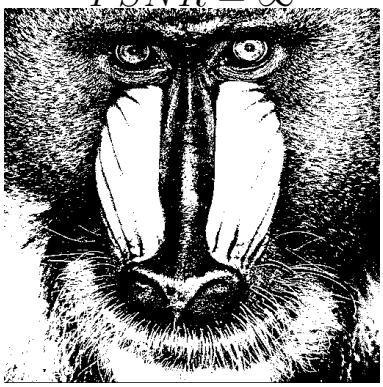
SC_4
$PSNR = \infty$



$C'_1 = C_1$
$PSNR = \infty$



$C'_2 = C_2$
$PSNR = \infty$



$C'_3 = C_3$
$PSNR = \infty$

$C'_4 = C_4$
$PSNR = \infty$

$S'_{1234} = S'_{2345} = S$
$PSNR = \infty$

Рис. 3: Результаты работы для цветного изображения

В языке JavaScript изначально не предусмотрена возможность работы с длинной арифметикой, поэтому в проекте используется библиотека с открытым исходным кодом "Big.js". Она предоставляет интерфейс Big, со структурой для хранения информации о числе и методами, реализующими базовый набор арифметических операций.

Основными функциями являются encrypt и recover. Они представляют собой работу алгоритмов 1 (ссылка) и 2(ссылка) соответственно. Рассмотрим их подробнее.

В функции encrypt входными параметрами являются:

1. sharesNum - число участников схемы;
2. threshold - пороговое число участников для восстановления секрета
3. secretPixels - одномерный массив со значениями пикселей секретного изображения;
4. covers - массив из sharesNum картинок прикрытия, представленных в форме одномерного массива;
5. TH - параметр, подробнее ((ССыылка)).

Изображения представляются в виде одномерного массива для удобства работы с индексами в случае работы с RGB и RGBA форматами.

Выходные данные:

1. modifiedCovers - доли, отправляемые участникам;
2. m - массив приватных чисел, соответствующих каждому из учатников;
3. T, p - публичные для всех участников числа.

В теле encrypt используются следующие функции:

• checkSizes - проверка размеров секретного изображения и картинок прикрытия – для серого изображения $L_S = L_{C_i}$, для RGB и RGBA – $L_S = \frac{1}{4} L_{C_i}$;

- MRandomize - выбор подходящих значений m_i на шаге 1 (ссылка);
- calcConsts - подсчет T, M, N, P;
- calcY - подсчет y на шаге 6 (ссылка);
- q - проверка условия на шаге 7 (ссылка).

В функции recover входные параметры совпадают с выходными данными в encrypt, за исключением количества долей и соответстующих им приватных чисел m_i.

Выходные данные:

1. coversRecovered - массив восстановленных картинок прикрытия, представленных одномерными массивами;
2. secretRecovered - восстановленное секретное изображение.

В теле recover используются:

- binThreshold - восстановление картинок изображения используя предельное(пороговое) значение;
- inverse - нахождение обратного элемента по модулю;
- CRTSolver - решение системы уравнений Китайской теоремы об остатках (ссылка).

Подробнее с результатами работы можно ознакомиться в репозитории проекта: (ссылка на гитхаб) Или скачать библиотеку на npm: «riss-color» (ссылка на npm)

Глава 5. Написание сайта, использующего библиотеку

Для демонстрации работы библиотеки было написано минимальное веб приложение. Оно позволяет протестировать работу библиотеки в браузере и на сервере, сравнить производительность. При написании сайта использована библиотека «React» (ссылка на оф документацию). Она позволяет сократить время на разработку и написанный на ней код легче поддерживать. Так же была использована библиотека «image-js», призванная унифицировать работу с изображениями, сокращая объем кода и потенциальное количество ошибок.

Веб приложение состоит из четырех компонентов:
App – главный компонент, отвечающий за рендер приложения,
CoversComponent – рендер картинок прикрытия и сохранение состояния, SharesComponent – рендер долей и сохранение их состояния, RenderImages – рендер произвольного числа изображений, сохраненных в состоянии

Функция rgbaToGrayscale - переводит картинку из формата RGBA в оттенки серого с помощью метода светимости:

$$GrayPixel = 0.3 * Red + 0.59 * Green + 0.11 * Blue$$

greyToBinary переводит серую картинку в бинарный формат. Эти функции решают проблему поиска подходящих картинок прикрытия и переводят их в нужный формат. Так же на сайте присутствует переключатель, который делает из загруженной секретной цветной картинки серую, позволяя снизить размер долей, если в сценарии приложения не важен цвет. RISS отвечает за подсчет одноименной метрики(ссылка) для изображений.

Интерфейс сайта представлен на Рисунке (ссылка). Сравнение производительности операции разделения секрета на клиенте и сервере представлены на Рисунке (ссылка).

Заключение

В ходе работы были изучены различные методы разделения секрета, написана библиотека для языка JavaScript, реализующая работу алгоритма обратимого разделения секретного изображения с картинками прикрытия. Так же было создано веб приложение, позволяющее продемонстрировать и протестировать работу библиотеки в реальных сценариях применения. На текущий момент, это единственная open-source библиотека JavaScript для разделения секретного изображения. В будущем планируется написать документацию и дополнить библиотеку пакетом Typescript, добаляющем статическую типизацию, для исключения возможных ошибок и работы с веб-приложениями, использующими этот пакет.

Список использованных источников

[1] Shamir, A. (1979). How to share a secret. Communications of the ACM, 22(11), 612-613.

[2] Salehi, S., & Balafar, M. A. (2015). An investigation on image secret sharing. International Journal of Security and its Applications, 9(3), 163-190.

[3] Yan, X., Lu, Y., Liu, L., & Song, X. (2020). Reversible image secret sharing. IEEE Transactions on Information Forensics and Security, 15, 3848-3858.

[4] https://www.scopus.com/

[5]

[6]

[7]