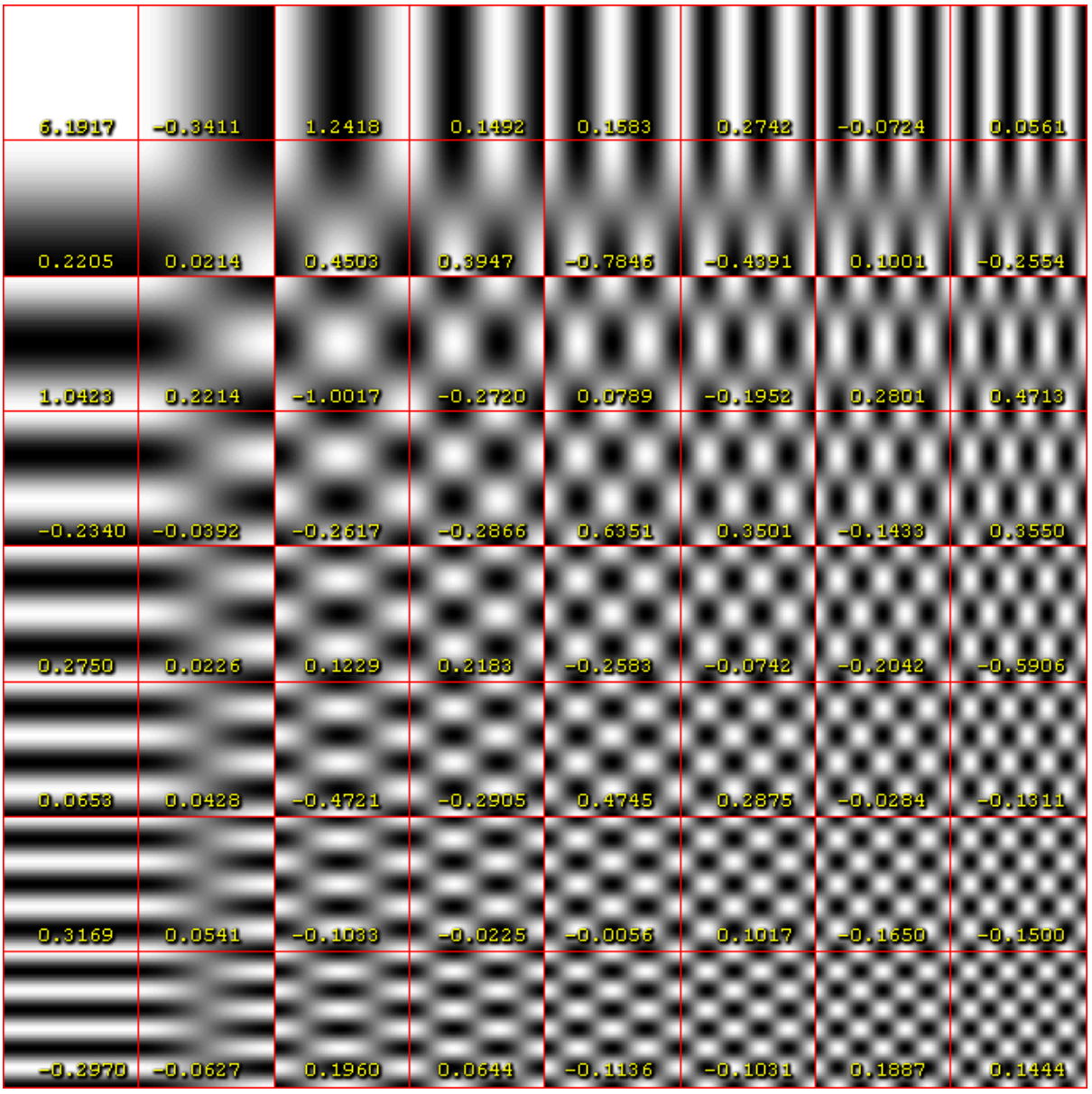
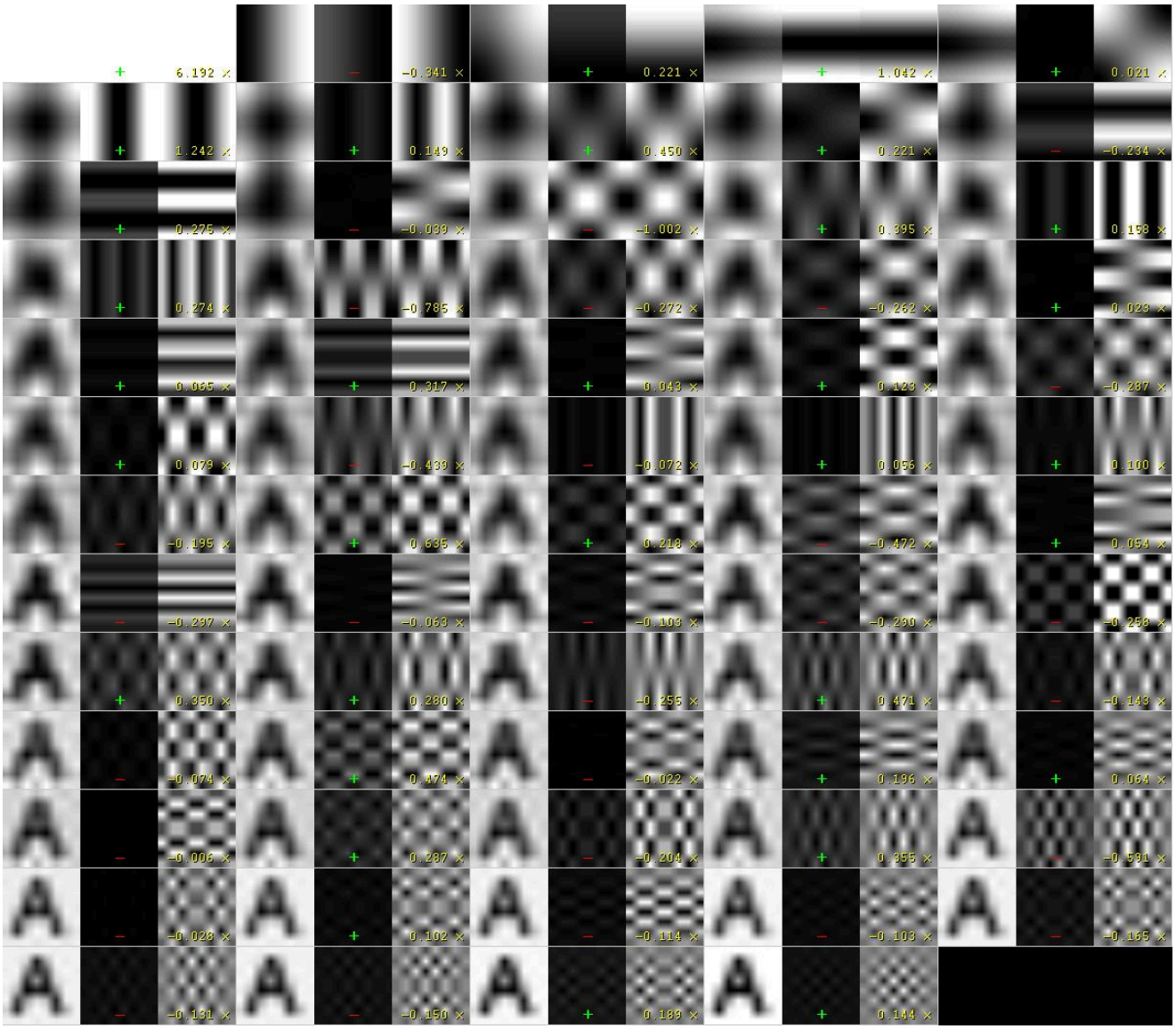
Алгоритм Коха-Жао

Дискретное косинусное преобразование

Для понимания алгоритма Коха-Жао нужно знать, что последовательность дискретных значений можно представить в виде суммы функций косинусов с различными периодами и различными коэффициентами. Таблица коэффициентов будет использоваться в дальнейшем. На следующем рисунке представлен пример комбинаций волн, представленных переходами цветов, с разными коэффициентами.



На следующем рисунке продемонстрировано, как при помощи комбинации волн с различными коэффициентами можно получить конечное изображение (слева показано получившееся изображение на данном шаге, справа - текущая волна и её коэффициент, по центру - текущая волна с применённым коэффициентом).



Формально ДКС является линейной обратимой функцией $f : \mathbb{R}^N \to \mathbb{R}^N$, где $\mathbb{R}^N$ обозначает множество рациональных чисел, или эквивалентно обратимой матрицей размерностью $N \times N$, и записывается следующим образом:

$$\zeta(x) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}$$

$$\Omega_{u,v} = \frac{\zeta(u) \times \zeta(v)}{\sqrt{2 \times N}} \times \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \cos(\frac{x \times v \times (2 \times x + 1)}{2 \times N}) \times \cos(\frac{x \times u \times (2 \times y + 1)}{2 \times N})$$

$$C_{x,y} = \frac{1}{\sqrt{2 \times N}} \times \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} (\zeta(u) \times \zeta(v) \times \Omega_{u,v} \times \cos(\frac{x \times v \times (2 \times x + 1)}{2 \times N}) \times \cos(\frac{x \times u \times (2 \times y + 1)}{2 \times N})$$

Приведённый пример является одной из многих форм ДКС и применяется кроме всего прочего в сжатии JPEG-изображений, которые также разбиваются на блоки 8×8 пикселей, после чего к блокам применяется ДКС, и в силу того, что блоки достаточно малы, чтобы в них в основном отсутствовали резкие переходы цветов, а получающиеся матрицы коэффициентов в случае однотонности блока содержат множество нулей по главной диагонали матрицы, такие матрицы коэффициентов хорошо сжимаются алгоритмом Хаффмана, что позволяет значительно уменьшить размер файла при незначительных потерях качества.

Шаги алгоритма

Алгоритм Коха-Жао для встраивания информации использует частотную область контейнера и заключается в относительной замене величин коэффициентов ДКП. Изображение разбивается на блоки размерностью 8×8 пикселей и к каждому блоку применяется ДКП. Каждый блок пригоден для записи одного бита информации.
Итак, алгоритм зашифрования будет следующим:

- разбиение изображения на блоки 8×8 пикселей;
- применение ДКП к этому массиву, и получаем массив коэффициентов размером 8×8;
- выбираем 2 коэффициента и высчитываем их разность по модулю;
- если разность меньше или равна P, то присваиваем первому коэффициенту положительное значение второго + константа, либо тоже самое но со знаком минус(это называется передача бита);
- если разность меньше или равна $-P$, то те же действия только для второго коэффициента;
- далее применяется обратное ДКП чтобы перевести наши коэффициенты обратно в пространственную область;
- после чего копируем новые значения цветов в изображение.
  Во время организации секретного канала абоненты должны предварительно договориться о двух конкретных коэффициентах ДКП из каждого блока, которые будут использоваться для скрытия данных.
  Таким образом, первичное изображение искажается за счет внесения изменений в коэффициенты ДКП, если их относительная величина не отвечает скрываемому биту. Чем больше значение P, тем стеганосистема, созданная на основе данного метода, является более стойкой к компрессии, однако качество изображения при этом значительно ухудшается.
  После соответствующего внесения коррекции в значения коэффициентов, которые должны удовлетворять неравенству, проводится обратное ДКП.
  Для извлечения данных в декодере выполняется аналогичная процедура выбора коэффициентов, а решение о переданном бите принимается в соответствии со следующим правилом:

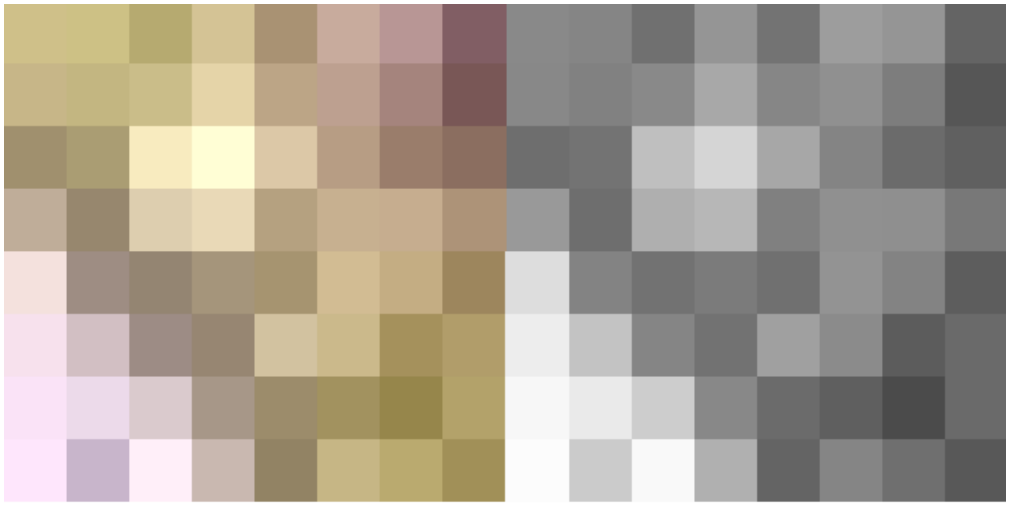
$$|\Omega_b(u_1, v_1)| > |\Omega_b(u_2, v_2)| \implies m_b^* = 0 |\Omega_b(u_1, v_1)| < |\Omega_b(u_2, v_2)| \implies m_b^* = 1$$

  Где Ω_b - массив коэффициентов, полученный в результате ДКП над b-ым блоком пикселей размерностью 8×8.

Пример использования алгоритма

Разберём алгоритм на примере одного блока 8×8, бита 1, $u_1 = 4$, $v_1 = 5$, $u_2 = 5$, $v_2 = 4$ и $P = 25$.

1. Возьмём блок пикселей размером 8×8 исходного изображения и выделим из него слой, отвечающий за яркость синего цвета



2. Применим ДКП к блоку пикселей "синего" слоя. При этом $\Omega_{0,0}$ содержит информацию о яркости всего сегмента, коэффициенты низкочастотных компонентов размещены ближе к левому верхнему углу, а высокочастотные - к нижнему правому. Ячейки, содержащие коэффициенты, которые будут использоваться для зашифрования и расшифрования бита, записанного в блок, выделены.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	36652.00	4479.18	-331.80	660.26	-48.08	1277.98	832.39	-349.08
1	-1540.05	-2532.52	-1539.78	45.82	53.49	437.04	-516.54	-344.04
2	436.77	858.09	231.24	-396.23	-572.14	-68.95	-56.88	-191.55
3	-760.93	-99.56	1323.31	1384.34	138.61	-691.65	-427.61	-162.13
4	59.40	-545.49	382.49	252.39	-334.00	437.41	207.42	-140.74
5	239.24	50.25	-29.86	-728.61	-462.71	-206.75	-20.90	32.96
6	367.09	-200.58	-556.88	78.06	139.57	-187.42	24.76	94.41
7	-357.65	219.01	9.64	-89.56	12.27	69.70	-17.94	-29.07

3. Вычислим разность абсолютных значений выделенных ячеек: $437.41 - |-462.71| = -25.3$. $-25.3 < -25$, значит, изменять значения коэффициентов нет необходимости. Если бы в данный блок нужно было зашифровать бит 0, $\Omega_{4,5}$ нужно было бы увеличить, а $\Omega_{5,4}$ уменьшить, чтобы $\Omega_{4,5} - |\Omega_{5,4}|$ было больше 25.

4. Применим обратное ДКП к матрице коэффициентов. Так как значения коэффициентов не изменялись, полученный блок значений яркости синего цвета не изменится.

   Для расшифровывания зашифрованного бита все действия выполняются в том же порядке, но на 3 шаге происходит сравнение коэффициентов $\Omega_{4,5}$ и $\Omega_{5,4}$, и если их разность меньше -25, то был зашифрован бит 1, а если больше 25, то 0. Шаг 4 при расшифровывании не выполняется.

   Исходное и конечное изображения представлены на следующих рисунках

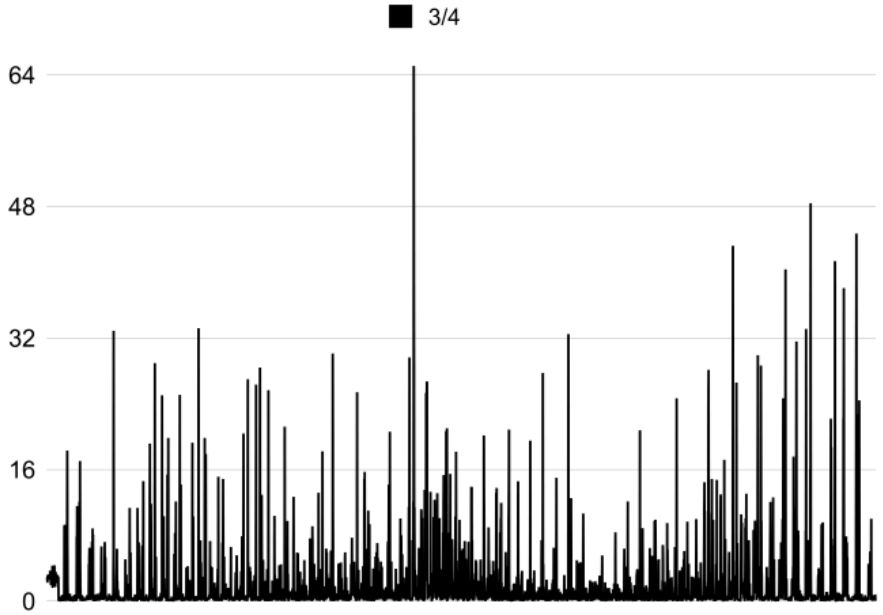
Стеганоанализ алгоритма

Основной целью стеганографического анализа (стегоанализа) является исследование стойкости схемы стеганографического встраивания к атакам различного типа. Традиционно формулируются три основные задачи стегоанализа. Первая состоит в обнаружении факта наличия встроенного сообщения. При обнаружении встроенного сообщения ставится задача вычисления его размера и расположения в контейнере. Третьей и самой сложной задачей является извлечение встроенного сообщения и его интерпретация без каких-либо данных о параметрах встраивания.
На сегодняшний день основные успехи стегоанализа связаны с решением первой задачи. Причём применяются преимущественно статистические методы исследования контейнера с встроенным сообщением. Все эти методы основаны на предположении о том, что встраивание сообщения вносит изменения в статистические характеристики контейнера, которые могут быть обнаружены на основе исследования различных распределений.
Так существует предположение о случайном характере распределения младших битов синей компоненты и на его основе применяют критерий χ-квадрат для задачи обнаружения стегановставки. Предложенный метод даёт хорошие результаты при равномерном заполнении контейнера. Для решения второй и третьей задачи статистических методов недостаточно и необходимо применять интеллектуальные алгоритмы. Например, существует метод анализа иерархий, что позволяет с высокой степенью точности определить размеры и положение встроенной информации.
Изменение среднечастотных компонент дискретного косинусного преобразования позволяет минимизировать визуальные эффекты встраивания. Встраивание в низкочастнотные компоненты приводит к заметному изменению фона изображения. Встраивание в высокочастное компоненты приводит к потере мелких деталей изображения.
Для решения задач стегоанализа будем исходить из того, что пороговое значение P должно иметь большое значение. В противном случае особенности изображения, используемого в качестве контейнера, могут приводить к ошибкам при извлечении данных.
На первом этапе необходимо выявить пары коэффициентов дискретного косинусного преобразования, используемые для встраивания информации. По аналогии с алгоритмом извлечения сообщения разобьём изображение-контейнер на блоки $C_{x,y}$ размером 8×8 пикселей. Выполним дискретное косинусное преобразование для каждого блока $C_{x,y}$ и найдём матрицы коэффициентов $\Omega_{u,v}$, которые также имеют размер 8×8. Выполним анализ элементов матриц $\Omega_{u,v}$. Для этого построим три последовательности:

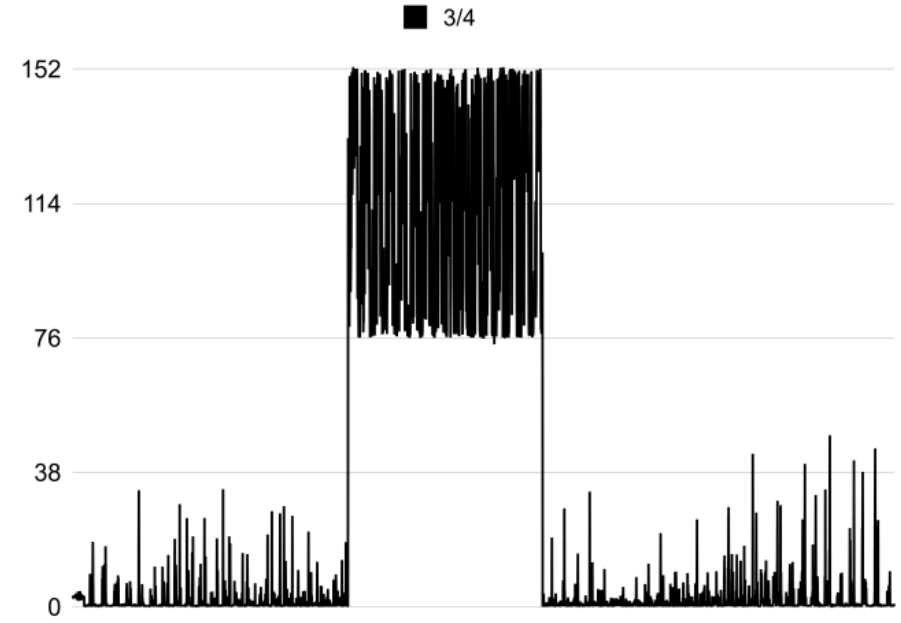
$$C_i(1) = ||\Omega_i[3,4]| - |\Omega_i[4,3]||, i = 0, \dots, N, C_i(2) = ||\Omega_i[3,5]| - |\Omega_i[5,3]||, i = 0, \dots, N, C_i(3) = ||\Omega_i|$$

Если встраивание было осуществлено в среднечастотную компоненту, то одна из этих последовательностей должна значительно измениться. Для каждой из последовательностей $C_i(j), i \in [0; N); j = 1, 2, 3$ проанализируем гистограмму

зависимости от номера блока i Встроенное сообщение приводит к появлению изменённого блока в виде ступенчатой зависимости. Причём высота ступени зависит от порогового значения P.
На следующем рисунке приведён пример гистограммы для последовательности без встроенного сообщения.



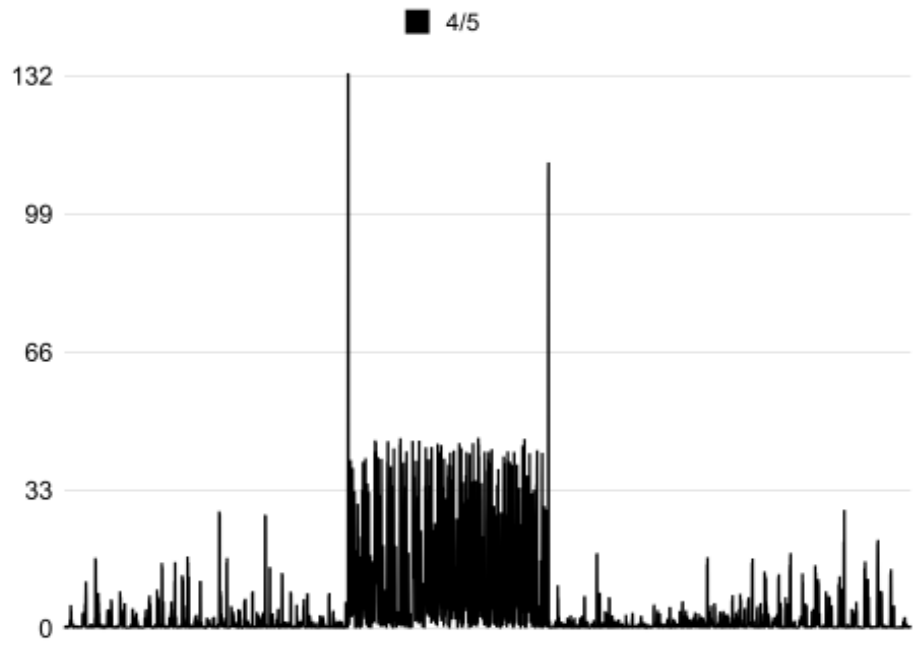
На следующем рисунке приведён пример гистограммы для последовательности со встроенным сообщением.



Таким образом, стегоанализ алгоритма Коха–Жао сводится к анализу зависимости последовательностей $C_i(j), i \in [0; N); j = 1, 2, 3$ и выявлению участка ступенчатых изменений. После обнаружения ступенчатых участков необходимо определить их границы. Для этого выполним численное дифференцирование зависимости $C_i(j), i \in [0; N); j = 1, 2, 3$ по i с использованием конечных разностей

$$dC_i = C_i(j) - C_{i-1}(j)$$

В точках ступенчатого изменения после дифференцирования будут наблюдаться высокие пики, соответствующие границе встраивания сообщения. Гистограмма $dC_i(j)$ для случая встроенного сообщения представлена на следующем рисунке.



Таким образом можно легко установить присутствие в изображении скрытого сообщения и границы встраивания. Стеганоанализ алгоритма стеганографического встраивания Коха-Жао, проведённый в данной статье, выявил неустойчивость к атаке анализа коэффициентов дискретного косинусного преобразования.

Заключение

Алгоритм Коха-Жао представляет собой инновационный подход к стеганографическому встраиванию информации в изображения с использованием дискретного косинусного преобразования (ДКП). Этот метод демонстрирует значительный потенциал для создания стойких к анализу схем стеганографического встраивания.

- Алгоритм использует блоки размером 8×8 пикселей, что позволяет эффективно встраивать информацию в изображения.
- Метод основан на относительной замене коэффициентов ДКП, что позволяет скрыть информацию от простого визуального анализа.
- Алгоритм демонстрирует хорошую стойкость к компрессии изображений, что делает его перспективным для применения в системах сжатия данных.
  Однако, как показал проведенный анализ, алгоритм Коха-Жао не является абсолютно стойким к атакам на основе анализа коэффициентов ДКП. Это ограничивает его применение в ситуациях, где требуется абсолютная стойкость.

Приложение

```
import numpy as np
import random
from skimage import io
from skimage.util import view_as_blocks
from scipy.fftpack import dct, idct

u1, v1 = 4, 5
u2, v2 = 5, 4
n = 8
P = 25
layer_to_modify = 2

def double_to_byte(arr):
    return np.uint8(np.round(np.clip(arr, 0, 255), 0))

def increment_abs(x):
    return x + 1 if x >= 0 else x - 1

def decrement_abs(x):
    if np.abs(x) <= 1:
        return 0
    else:
        return x - 1 if x >= 0 else x + 1

def abs_diff_coefs(transform):
    return abs(transform[u1, v1]) - abs(transform[u2, v2])

def valid_coefficients(transform, bit, threshold):
    difference = abs_diff_coefs(transform)
    if (bit == 0) and (difference > threshold):
        return True
    elif (bit == 1) and (difference < -threshold):
        return True
    else:
        return False

def change_coefficients(transform, bit):
    coefs = transform.copy()
    if bit == 0:
        coefs[u1, v1] = increment_abs(coefs[u1, v1])
```

```
        coefs[u2, v2] = decrement_abs(coefs[u2, v2])
    elif bit == 1:
        coefs[u1, v1] = decrement_abs(coefs[u1, v1])
        coefs[u2, v2] = increment_abs(coefs[u2, v2])
    return coefs

def embed_bit(block, bit):
    patch = block.copy()
    coefs = dct(dct(patch, axis=0), axis=1)
    while not valid_coefficients(coefs, bit, P) or (bit !=
retrieve_bit(patch)):
        coefs = change_coefficients(coefs, bit)
        patch = double_to_byte(idct(idct(coefs, axis=0), axis=1)/(2*n)**2)
    return patch

def embed_message(orig, msg):
    changed = orig.copy()
    blue = changed[:, :, layer_to_modify]
    blocks = view_as_blocks(blue, block_shape=(n, n))
    h = blocks.shape[1]
    for index, bit in enumerate(msg):
        i = index // h
        j = index % h
        block = blocks[i, j]
        blue[i*n: (i+1)*n, j*n: (j+1)*n] = embed_bit(block, bit)
    changed[:, :, 2] = blue
    return changed

def retrieve_bit(block):
    transform = dct(dct(block, axis=0), axis=1)
    return 0 if abs_diff_coefs(transform) > 0 else 1

def retrieve_message(img, length):
    blocks = view_as_blocks(img[:, :, layer_to_modify], block_shape=(n,
n))
    h = blocks.shape[1]
    return [retrieve_bit(blocks[index//h, index%h]) for index in
range(length)]



original = io.imread('./TUV0V.png')
test_message = [random.randint(0, 1) for i in range(int(400 * 400 / n /
n))]
```

```
images = [original]

changed = embed_message(original, test_message)
retrieved_message = retrieve_message(changed, len(test_message))
print('Retrieved message', retrieved_message)
print('Messages are equal:', test_message == retrieved_message)
images.append(changed)

io.imshow(np.hstack(images))
```