Ассоциативный механизм защиты данных

Ассоциативная защита – это симбиоз стеганографии и криптографии, изначально разработанный для защиты данных при анализе сцен [1]. Большая часть ранних исследований [2,3] была нацелена на управление защищенными картографическими базами данных, где каждый код объекта/координаты после представления цифр этого кода в виде бинарных отрезков, маскирования и рандомизации преобразуется в стегонтейнер, состоящий из k секций. Принцип формирования стегоконтейнера следующий.

На первом этапе создается пустой контейнер, длина которого – $L = k \times (9 \times n - 12)$ – определяется количеством бит бинарных матриц-эталонов, которые могут выступать в качестве существенных. Эти биты располагаются по внешнему контуру и внутреннему «зигзагу» соответствующих бинарных матриц (рисунок 1; а – $n = 3$, б – $n = 7$; где n – число столбцов бинарной матрицы-эталона).

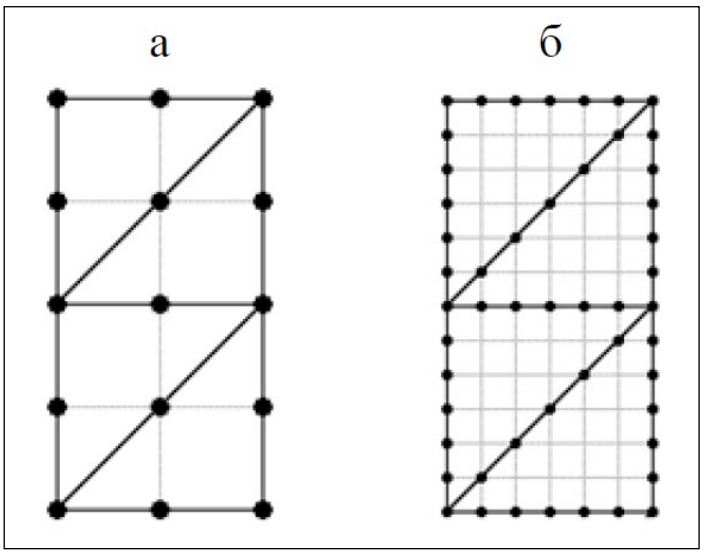

Рисунок 1 – Контур + «зигзаг»: а – $n = 3$, б – $n = 7$

Формирование контейнера происходит с использованием псевдослучайной последовательности (ПСП) [4]. После этого биты эталонов встраиваются в позиции контейнера, определяемые набором масок (секретным ключом) (см. рисунок 2).

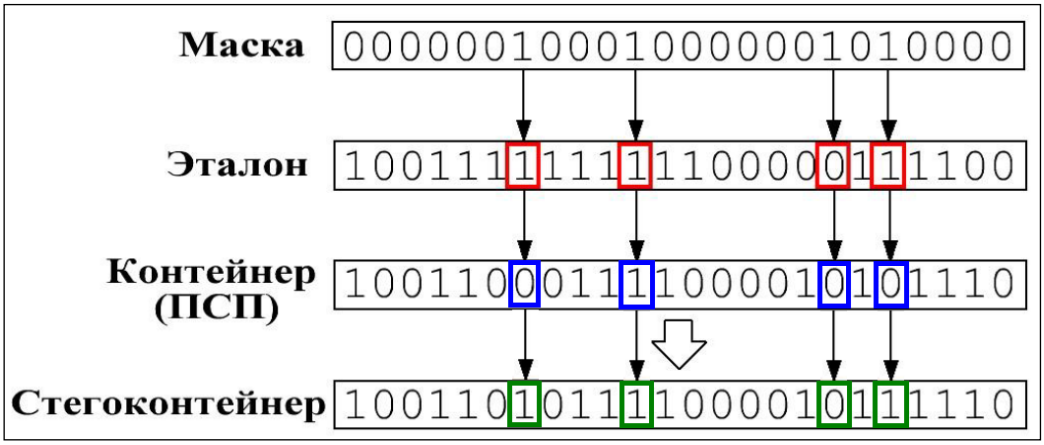

Рисунок 2 – Принцип формирования стегоконтейнера

Пример двух возможных вариантов наборов масок для десяти матриц-эталонов ($n = 3$), представленных в форме почтовых символов, приведен на рисунке 3.

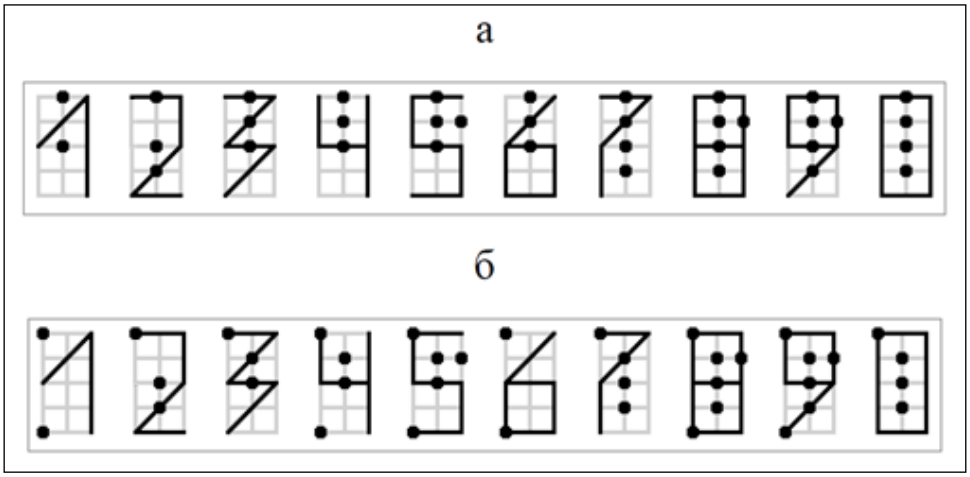

Рисунок 3 – Примеры вариантов маскирования при $n = 3$

Алгоритм формирования секретного ключа для исходного множества эталонов C представлен на рисунке 4 в виде диаграммы деятельности на языке UML. Рассмотрим логику дихотомического разбиения на каждой итерации алгоритма.

На начальном шаге алгоритма: $\{c_0\} \in D_1$, $\{c_1\} \in D_2$ $a_0 := c_0 \oplus c_1$, $a_1 := c_0 \oplus c_2$. Вектор a_0 хранит позиции, по которым эталоны множества D_1 отличаются от эталонов множества D_2. a_1 – это маска позиций, по которым c_0 отличается от проверяемого эталона c_{i+2}. Вычисление пересечения $a_3 := a_0 \,\&\, a_1$ с последующим присваиванием $a_0 := a_3$ (если $a_3 \neq \overline{0}$) на каждой итерации алгоритма позволяет сужать набор различий эталонов множеств D_1 и D_2.

Если $a_3 = \overline{0}$ (пересечения нет), то это означает, что c_0 отличается от эталонов множества D_2 в одних позициях, а от c_{i+2} – в совершенно других позициях. Для случая, когда c_0 отличается от c_1 только в первой позиции ($a_0 = 1000$), а от c_2 – только во второй позиции ($a_1 = 0100$), получаем отсутствие пересечения позиций различий ($a_3 = 0000$). Раз позиции различий не пересекаются, значит в первой позиции эталоны c_1 и c_2 имеют разные значения, а c_0 и c_2 – одинаковые значения. В результате эталон c_2 добавляем в множество D_1 вместе с c_0. Рассмотрим пример:

$c_0 = 0110$ (опорный эталон);

$c_1 = 1110 \rightarrow a_0 = 1000$ (различие в первой позиции);

$c_2 = 0010 \rightarrow a_1 = 0100$ (различие во второй позиции);

$a_3 = 0000$ (нет общих позиций различий);

$\{c_0, c_2\} \in D_1; \{c_1\} \in D_2$.

Если $a_3 \neq \overline{0}$ (есть общие позиции различий). Это означает, что существуют позиции, где c_0 одновременно отличается и от эталонов множества D_2, и от c_{i+2}. Для случая, когда c_0 отличается от c_1 в первой позиции ($a_0 = 1000$), а от c_2 – в первой и второй позициях ($a_1 = 1100$), то c_1 и c_2 имеют одинаковое значение в первой позиции (противоположное значению в этой позиции эталона c_0). Наличие общей позиции различия указывает, что c_2 следует отнести к множеству D_2. Обновляя $a_0 := a_3$, мы оставляем только общие позиции различий – это сужает набор битов, по которым будем различать оставшиеся эталоны. Приведем пример:

$c_0 = 0110$ (опорный эталон);

$c_1 = 1110 \rightarrow a_0 = 1000$ (различие в первой позиции);

$c_2 = 1010 \rightarrow a_1 = 1100$ (различие в первой и второй позициях);

$a_3 = 1000$ (есть общая позиция различия);

$\{c_0\} \in D_1; \{c_1, c_2\} \in D_2$.

Здесь хорошо прослеживается суть алгоритма: эталоны исходного множества сопоставляются с опорным эталоном c_0 и распределяются на основе анализа позиций различий по множествам D_1 и D_2.

После анализа всех эталонов случайный выбор бита различия из a_0 определяет дихотомальный бит эталонов множества C. Выявление последующих битов различий продолжается рекурсивным выполнением алгоритма для сформированных множеств D_1 и D_2. Формируемые дихотомальные биты эталонов на каждом этапе работы алгоритма накапливаются в массиве M. Работа завершается, если мощности входных множеств для последующих рекурсивных вызовов алгоритма равны единице. Итоговый массив M определяет секретный ключ (набор масок).

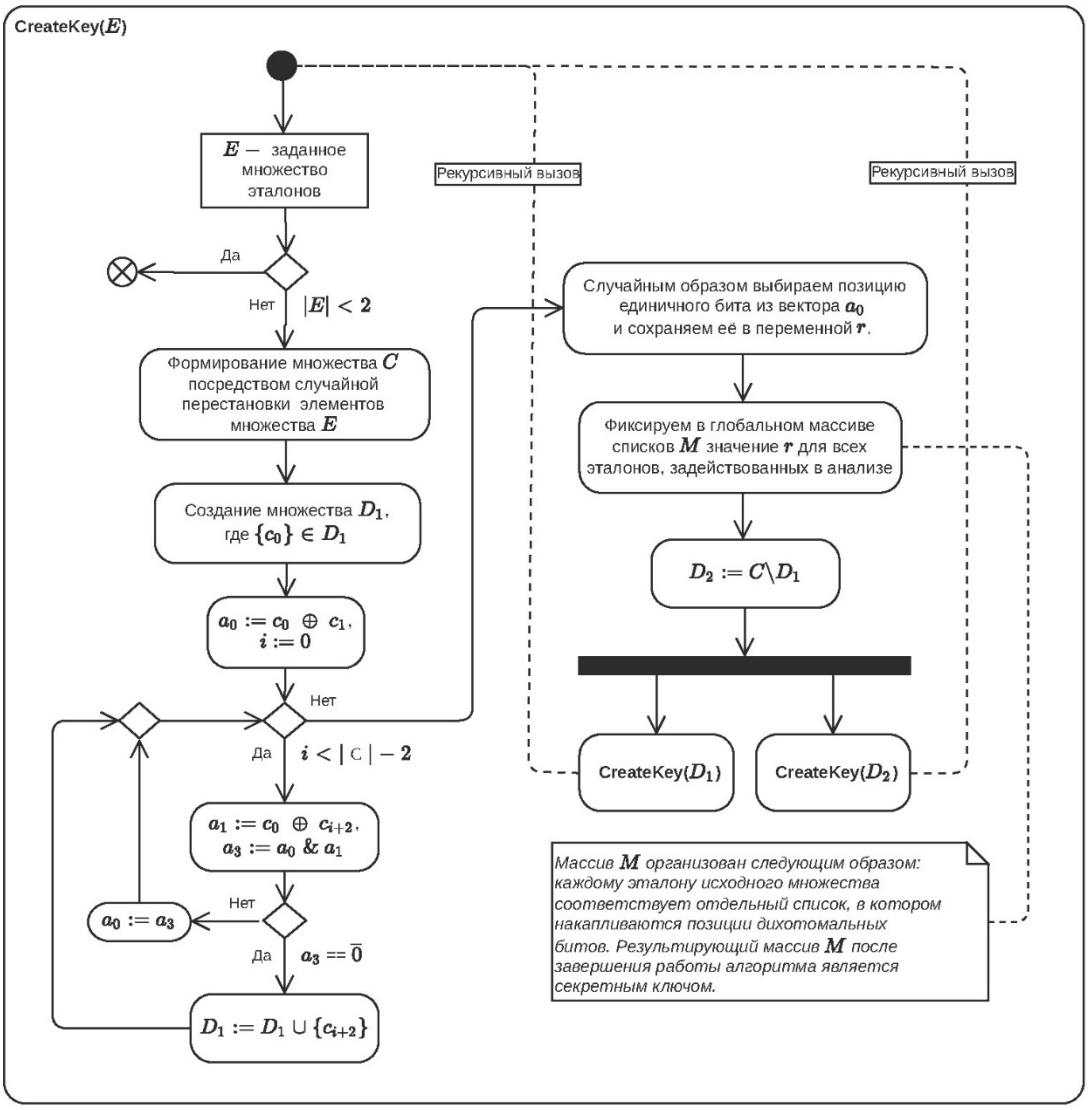

Рисунок 4 – Алгоритм формирования секретного ключа

Практическое использование ассоциативной стеганографии не ограничивается защитой картографических данных – сфера ее применения значительно шире [5-7]. Возможность ассоциативной защиты любой цифровой информации обеспечивается универсальным принципом кодирования: любой байт можно представить трехразрядным десятичным числом (от 000 до 255), поставив в соответствие каждой цифре скрываемого кода определенный эталон из набора десяти матриц. В случае использования 16 эталонов любому байту можно сопоставить двухсекционный стегоконтейнер, где каждая секция кодирует один полубайт (4 бита).

Таким образом, описанный механизм позволяет обеспечить защищенную передачу и хранение данных, где стойкость защиты определяется сложностью восстановления секретного ключа. Дихотомическое разбиение множества эталонов обеспечивает экспоненциальный рост вариантов маскирования, что делает метод устойчивым к атакам перебора при достаточной длине ключа.

Более детальное изложение механизма ассоциативной защиты данных представлено в диссертации [8].

Литература

1. Duda R.O., Hart P.E., Stork D.G. Pattern classification and scene analysis. New York: Wiley, 1973. V. 3. P. 731-739.

2. Raikhlin V.A., Vershinin I.S., Gibadullin R.F., Pystogov S.V. Reliable Recognition of Masked Binary Matrices. Connection to Information Security in Map Systems // Lobachevskii Journal of Mathematics, 2013. V. 34, №4. P. 319-325.

3. Raikhlin V.A., Gibadullin R.F., Vershinin I.S., Pystogov S.V. Reliable recognition of masked cartographic scenes during transmission over the network // 2016 International Siberian Conference on Control and Communications (SIBCON). IEEE, 2016. P. 1-5.

4. Tian X., Benkrid K. Mersenne Twister Random Number Generation on FPGA, CPU and GPU // 2009 NASA/ESA Conference on Adaptive Hardware and Systems. San Francisco, CA, USA, 2009. P. 460-464.

5. Raikhlin V.A., Vershinin I.S., Gibadullin R.F. The Elements of Associative Steganography Theory // Moscow University Computational Mathematics and Cybernetics, 2019. V. 43, №1. P. 40-46.

6. Гибадуллин Р.Ф., Гашигуллин Д.А., Вершинин И.С. Разработка декоратора StegoStream для ассоциативной защиты байтового потока // Моделирование, оптимизация и информационные технологии. 2023. Т. 11, № 2(41). С. 22-23.

7. Гибадуллин Р.Ф., Вершинин И.С., Глебов Е.Е. Разработка приложения для ассоциативной защиты файлов // Инженерный вестник Дона. 2023. № 6(102). С. 118-142.

8. Вершинин И.С. Элементы конструктивной теории ассоциативной защиты информации при ее хранении и передаче. // Докторская диссертация. Казанский (Приволжский) федеральный университет. 2025. URL: https://kpfu.ru/dis_card?p_id=3896