

ОБРАБОТКА ИЗОБРАЖЕНИЙ, РАСПОЗНАВАНИЕ ОБРАЗОВ

МОДЕЛИ СХЕМ РАЗДЕЛЕНИЯ СЕКРЕТА В СИСТЕМАХ ПЕРЕДАЧИ ВИДЕОИНФОРМАЦИИ

Сагайдак Д.А., Файзуллин Р.Т.

Омский государственный технический университет

Аннотация

В данной статье предложены два варианта реализации схем разделения секрета на неравные доли с принципиально малой частью секрета для организации защищённого хранения и обработки видеoinформации в центрах обработки данных, в основе которых лежит RGB-представление (аббревиатура английских слов Red, Blue, Green – красный, зелёный, синий) каждого из фреймов видеосигнала. Для каждой предложенной схемы выполнена программная реализация в среде программирования Borland C++ Builder 6.

Ключевые слова: центр обработки данных (ЦОД), разделение секрета, разделение видеоданных, представление в формате RGB, пиксельное представление изображения, фрейм, дельта-код Элиаса, префикс, недоверие к ЦОД, мафиозная атака.

Введение

Увеличение скорости передачи данных и пропускной способности каналов связи в настоящее время все чаще позволяет осуществлять потоковую передачу данных, зачастую такими данными является медиа-контент (аудио-, видеоданные). Для передачи потоковой видеoinформации используются технологии сжатия и буферизации данных (двойная, тройная, z-буферизация), позволяющие передавать видеoinформацию в режиме реального времени с помощью протоколов потокового вещания, например, таких как: RTP, RTSP, RTCP, SIP, H.323 и другие [1–4].

Повышение качества видеоизображения, возобновление интереса к стереооптическому видео (трёхмерное видео), появление видео высокого разрешения/чёткости (HDTV) – всё это способствует увеличению объёмов медиаконтента. В настоящее время для хранения объёмного медиаконтента всё чаще используются серверы дата-центров или так называемые центры обработки данных (ЦОД) и облака [5]. Использование услуг ЦОД способствует снижению общего числа серверного оборудования непосредственно у самих клиентов ЦОД, а также снижает расходы на их техническую поддержку. Но по-прежнему остаётся актуальным вопрос обеспечения конфиденциальности передаваемой на хранение в ЦОД информации. Даже поддержанные лицензиями уровни надёжности и меры обеспечения защиты информации в ЦОД не могут убедить клиентов в их надёжности, так как угроза мафиозной атаки [6] очень высока. Ввиду этого клиенты ЦОД, в качестве которых могут выступать как обычные пользователи, так и крупные организации, размещающие свой медиаконтент в целях достижения конфиденциальности передаваемых видеоданных, осуществляют самостоятельное шифрование этих данных. В этом случае выбор алгоритма шифрования возлагается на клиентов ЦОД, которые часто не являются специалистами по защите информации и не могут обеспечить должный уровень защиты. Таким образом, получается противоречие. Владелец видеoinформации предполагает хранить её на сервере ЦОД, но он не доверяет

шифрованию данных, осуществляемому на стороне ЦОД, и не имеет должной квалификации, чтобы самостоятельно криптографически защитить информацию на своей стороне.

Также следует отметить, что в случае осуществления потоковой передачи видеоданных с серверов ЦОД возможности большинства алгоритмов шифрования могут быть ограничены ввиду их недостаточного быстродействия, так как в основе большинства из них лежит блочное шифрование, которое оказывает влияние на скорость шифрования, особенно это заметно, когда используемые алгоритмы шифрования реализуются программным обеспечением. Для решения проблем, связанных с увеличением скорости шифрования, зачастую прибегают к построению схем шифрования, основанных на поточном шифре, но поточные шифры могут быть подвержены корреляционным атакам.

Сейчас наибольшее распространение получили видеоданные в формате семейства MPEG ввиду своей универсальности. Но следует отметить, что в самом семействе форматов сжатия MPEG не предусмотрено решений по обеспечению конфиденциальности, так как процесс выполнения одновременного сжатия и шифрования довольно сложный. Поэтому, зачастую прибегают к использованию некоторой последовательной схемы, состоящей из сжатия видеоданных и последующего их шифрования. Основным недостатком данной схемы являются высокие требования к производительности оборудования в связи с необходимостью осуществлять шифрование больших объёмов информации. Тем более что в сетевых приложениях, например, при осуществлении вещания с сервера ЦОД, требования по производительности сервера, когда он обслуживает тысячи клиентов, выполнить практически невозможно. Критична по требованию к производительности оборудования и сторона, на которой осуществляется дешифрование принимаемых видеоданных (клиентская сторона ЦОД). Часто на практике применяются следующие методы защиты видеоданных в формате семейства MPEG: метод случайной перестановки коэффициентов дискретного косинусного

преобразования (ДКП), метод селективного шифрования, метод шифрования на основе изменяемых кодовых таблиц, которые также не лишены недостатков, основные из которых описаны в [7].

Постановка задачи

На основании вышесказанного в подобной ситуации представляется возможным использовать схемы разделения секрета [8], когда на сервере ЦОД осуществляется не хранение информации как таковой, а некоторых данных. Ввиду того, что технические возможности ЦОД позволяют хранить большие объёмы данных, то размер части секрета, хранимого у пользователя для воспроизведения потока видеоданных с сервера ЦОД, должен быть мал. Казалось бы, эту малую часть можно рассматривать как ключ, хранящийся у клиента, но принципиальное отличие от схем шифрования заключается в том, что, по всей видимости, здесь можно добиться строгих результатов о невозможности восстановления информации только по большей части данных. Следует учесть, что декомпозиция и композиция информации осуществляется на клиентской стороне схемы, поэтому требуется применение максимально простых и эффективных алгоритмов, работающих «на лету».

Сейчас предложено достаточно пригодных схем разделения секрета, например такие как [9] и [10], но в данных работах не делается принципиального акцента на размер части секрета. Отсутствуют работы по приложению схем разделения секрета для обеспечения конфиденциальности видеоданных, когда те хранятся на серверах ЦОД. Поэтому в данной работе будут предложены два варианта схем разделения секрета на неравные доли, где часть секрета будет являться лишь малой частью передаваемой видеoinформации.

Описание первой схемы разделения секрета

Пусть имеется некий несжатый поток видеoinформации расширением $M \times N$. Первоначально осуществляется деление данного потока видеoinформации на фреймы (от англ. frame – «кадр»), затем для каждого фрейма осуществляется построчное чтение пикселей (от 1 до M). Для каждого прочитанного пикселя каждой строки находится его представление в формате RGB, то есть три десятичных числа от 0 до 255, значения которых переводятся в двоичную систему исчисления и записываются последовательно друг за другом. В результате получается битовая последовательность T , состоящая из 24 бит.

$$T = [r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 g_1 g_2 g_3 g_4 g_5 g_6 g_7 g_8 b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8],$$

где $r_1 - r_8$, $g_1 - g_8$, $b_1 - b_8$ – порядковые номера 24-битовой последовательности.

Над каждой такой последовательностью выполняется следующая перестановка:

$$\bar{T} = [r_1 g_1 b_1 r_2 g_2 b_2 r_3 g_3 b_3 r_4 g_4 b_4 r_5 g_5 b_5 r_6 g_6 b_6 r_7 g_7 b_7 r_8 g_8 b_8].$$

Далее последовательность $\bar{T}$ разбивается на блоки по четыре бита, которые являются дробной частью десятичных чисел a_i :

$$a_1 = 0, r_1 g_1 b_1 r_2;$$

$$a_2 = 0, g_2 b_2 r_3 g_3;$$

$$a_3 = 0, b_3 r_4 g_4 b_4;$$

$$a_4 = 0, r_5 g_5 b_5 r_6;$$

$$a_5 = 0, g_6 b_6 r_7 g_7;$$

$$a_6 = 0, b_7 r_8 g_8 b_8.$$

Таким образом, для каждой пиксельной строки изображения создаётся массив A размерностью $6 \cdot M$, состоящий из полученных a_i для каждого пикселя пиксельной строки изображения.

Далее находится среднее значение массива A :

$$sr_A = \frac{\sum_{i=1}^{6 \cdot M} a_i}{6 \cdot M} \quad (1)$$

и формируется новый массив $\bar{A}$ размерностью $6 \cdot M$, состоящий из элементов $\bar{a}_i = sr_A - a_i$, где $i = 1..6 \cdot M$, sr_A – значение, полученное в (1).

Далее находится новый массив F размерностью $6 \cdot N$, состоящий из элементов:

$$f_i = -a_{i-1} + 2a_i - a_{i+1}, \text{ если } i \neq 1 \text{ и } i \neq 6 \cdot M;$$

$$f_i = -a_{i-1} + 2a_i - a_1, \text{ если } i = 1 \text{ и } i \neq 6 \cdot M;$$

$$f_i = -a_{6 \cdot M} + 2a_i - a_{i+1}, \text{ если } i = 6 \cdot M \text{ и } i = 1.$$

Из правых частей приведённых выше уравнений можно составить квадратную и симметричную относительно главной диагонали матрицу вида:

$$H = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 & 0 & -1 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & 0 & 0 & 0 & -1 & 2 \end{pmatrix}. \quad (2)$$

Все полученные элементы массива F меньше 1, т.е. $f_i < 1$. Из проведённых экспериментов получено, что для дальнейшего восстановления секрета необходимо производить вычисление элементов массива F с точностью до шести-девяти знаков после запятой.

В различных случаях в качестве ключа могут выступать:

а) последовательность из знаков *всех* элементов массива F , тогда в данном случае открытой информацией является массив $\bar{F}$ размерностью $6 \cdot M \cdot N$, состоящий из элементов $|f_i|$. Пользователь получит ключевую последовательность из знаков элементов массива F , которая сжимается без потерь по принципу:

$$-- \rightarrow -;$$

$$-+ \rightarrow +-;$$

$$+- \rightarrow ++-;$$

$$++ \rightarrow +++;$$

б) последовательность из знаков *некоторых* элементов массива F , тогда открытой информацией является массив $\bar{F}$ размерностью $6 \cdot M \cdot N$. Пользователь получит ключевую последовательность, состоящую из изъятых знаков и мест их изъятия;

в) последовательность, состоящая из средних значений sr_A , полученных по (1).

На рис. 1а приведено исходное изображение фрейма, а на рис. 1б – изображение фрейма, когда атакующий попытается восстановить исходное изображение фрейма по элементам $|f_i|$ массива $\bar{F}$. Восстановление осуществляется «в лоб», т.е. путём непосредственного перевода элементов $|f_i|$ массива $\bar{F}$ в значения пикселей в формате RGB.



Рис. 1. Исходное изображение фрейма (а), изображение фрейма при попытке восстановления по элементам массива $\bar{F}$ (б)

Из рис. 1б видно, что при попытке восстановить исходное изображение фрейма проглядываются некоторые резкие границы изображения. Чтобы избежать проявления этих границ, возможно осуществлять преобразование исходного фрейма в несколько раундов, зачастую достаточно двух раунда. Полученное изображение после первого раунда поворачивается на 90 градусов (по часовой или против часовой стрелки), и осуществляется второй раунд. Также возможно вместо осуществления второго раунда выполнить некоторую перестановку строк и столбцов, описанную в [11]. На рис. 2 приведено изображение, полученное при попытке восстановить исходное изображение фрейма при наличии у атакующего массива $\bar{F}$, состоящего из элементов $|f_i|$, после двух раундов преобразования исходного фрейма без осуществления какой-либо перестановки строк и столбцов изображения после первого раунда преобразования.

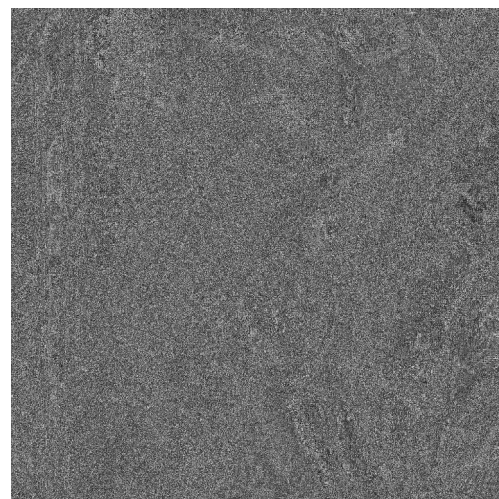


Рис. 2. Изображение фрейма при попытке восстановления по элементам массива $\bar{F}$ после двух раундов преобразования исходного изображения фрейма

Предложенный метод хорош для многоцветных изображений, но если на входе будет чёрно-белое изображение с чётко выраженными границами, как на рис. 3а, то в итоге преобразований будет прослеживаться данный переход.

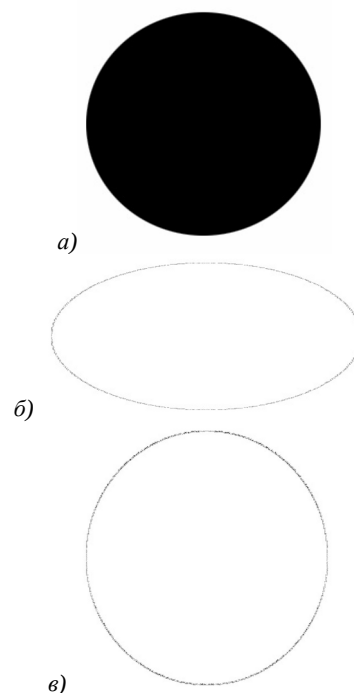


Рис. 3. Исходное изображение фрейма (а), изображение фрейма после первого раунда преобразования (б), изображение фрейма после двух раундов преобразования (в). Фон изображений (б) и (в) инвертирован (вместо белого фона подразумевается чёрный фон)

Восстановление исходного изображения

Пусть пользователь имеет ключ S и данные F . Так как требуется быстрое восстановление секрета, то предполагается использование метода Якоби для матрицы H , приведённой в (2). Для данной матрицы нахо-

дятся собственные числа λ_i и собственные векторы v_i , тогда секрет можно восстановить по следующему принципу: $F\bar{A} = S$, где $\bar{A}$ – искомый секрет.

$$1. S = \sum_{i=1}^{6-M} \beta_i \cdot v_i \Rightarrow \beta_i = (S, v_i), \text{ т.е. выполняется}$$

скалярное произведение векторов S и v_i , где S – ключ, представляющий из себя числовой вектор, β_i – скалярное произведение вектора S , на собственные векторы v_i матрицы H .

$$2. \text{Находим элементы вектора } \bar{A} = \sum_{i=1}^{6-M} \frac{\beta_i}{\lambda_i} \cdot v_i.$$

Элементы исходного массива A равны $a_i = sr_A - \bar{a}_i$. Выполнив перестановку, обратную перестановке, описанной выше, т.е. $\bar{T} \Rightarrow T$, и осуществив преобразование значений пикселей из двоичной системы счисления в десятичную, удастся восстановить исходное изображение.

Описание второй схемы разделения секрета

Рассмотрим задачу хранения большого числа массивов данных, длины записи которых существенно различаются. Пусть даны n битовых векторов $A_1, \dots, A_n$, размерности которых равны $M_1, \dots, M_n$ соответственно и дисперсии M_i распределены равномерно в достаточно большом интервале.

В этом случае возникает проблема экономичной записи данных, которая в настоящее время решается различными способами: шардинг [12], т.е. грубое физическое разделение данных по различным носителям данных; введение различных типов данных наподобие CHAR и VARCHAR; разделение данных маркерами; создание дисковых массивов типа RAID 0. Но если M_i варьируются от 10^3 до 10^{10} , а n изменяется, то отведение равных областей памяти для каждого A_i неэффективно, а разделение данных специальной строкой бит (маркером) неэффективно по времени поиска этого маркера, и нет никакой гарантии, что выбранная в качестве маркера строка не встретится ни в одном из A_j [13].

Рассмотрим алгоритм, представляющий собой примитивизацию дельта-кода Элиаса (универсальный код для кодирования целых чисел, разработанный Питером Элиасом) [14]. Первые l бит заполним нулями, где l – это длина записи числа n в двоичной системе счисления, далее осуществляется сама запись числа n в двоичной системе счисления. Например, пусть даны $n=3$ битовых векторов, тогда запись числа n в двоичной системе счисления равна 11, тогда $l=00$. Следовательно, на первом этапе получается следующая числовая последовательность – 0011. Далее m_i бит заполняются нулями, где m_i – это число бит, необходимых для записи длины вектора A_i в двоичной системе счисления. Например, в предыдущем примере $n=3$, следовательно, имеется три би-

товых вектора A_1, A_2, A_3 , пусть $A_1=111011$, $A_2=10111$, $A_3=101$, тогда размерности этих битовых последовательностей равны $M_1=6$, $M_2=5$, $M_3=3$ соответственно, а $m_1=000$, $m_2=000$, $m_3=00$. На втором этапе получится следующая последовательность – 0001100001010011. Третьим этапом формирования последовательности является последовательная запись самих векторов A_1, A_2, A_3 . Например, для приведённого выше примера получится следующая последовательность:

0011000110000101001111101110111101.

Обратим внимание на то, что, зная диапазон возможных изменений A_i , можно записывать A_i в $m_i + d_i$ позициях, предвзя или дополняя нулями значащие цифры A_i . Это позволяет легко перезаписывать и дописывать массивы и их новые значения, не усложняя структуру данных.

Также возможно использовать предложенный выше метод в качестве основы схемы разделения секрета для потока видеoinформации, и мы попытаемся построить алгоритм, не требующий значительных вычислительных ресурсов.

Пусть имеется поток определенного («телевизионного») формата 720×576 пикселей 25 кадров в секунду в формате RGB (в дальнейшем будет осуществляться преобразование видеопотоков стандартных форматов: 720×576 , 640×480 , 352×288 (CIF – Common Interchange Format), 176×144 (QCIF – Quartered Common Interchange Format)), т.е. размерность изображения является известной и выбирается из одного из стандартов. Осуществляется разбиение данного потока на фреймы. Производится построчное чтение пикселей фрейма, затем для каждого пикселя строки находятся его значения в формате RGB в двоичной системе счисления (размерностью 24 бита, т.е. по 8 бит для каждого цвета) и записываются последовательно друг за другом в одну строку, создавая последовательность, состоящую из нулей и единиц. Каждая такая достаточно длинная последовательность строки прочитанных пикселей разбивается на n битовых векторов $A_1, A_2, \dots, A_n$ разных размерностей $M_1, \dots, M_n$.

В качестве примера: для генерации размерностей $M_1, \dots, M_n$ битовых векторов $A_1, A_2, \dots, A_n$ в программе, реализующей описываемый метод, используется функция *RandomRange()*, встроенная в среду программирования Borland (C++, Delphi), позволяющая осуществлять генерацию псевдослучайных чисел в пределах заданного диапазона (например, от 20000 до 30000) и удовлетворяющая общим требованиям к датчикам псевдослучайных чисел, описанным в стандарте FIPS 140-1 (Federal Information Processing Standards) [15]. Понятно, что можно использовать и другие генераторы псевдослучайных чисел, удовлетворяющие более важным требованиям.

В свою очередь, над полученными битовыми векторами $A_1, A_2, \dots, A_n$ осуществляются следующие операции:

$$X_1 = A_2 \otimes A_3, \quad X_2 = A_3 \oplus A_4, \dots,$$

$X_{n-2} = A_{n-1} \oplus A_n, \quad X_{n-1} = A_n \oplus A_1$ (где « $\oplus$ » – побитовое сложение по модулю 2), полученные битовые векторы $X_1, X_2, \dots, X_{n-1}$ записываются последовательно друг за другом. Следует заметить, что битовые векторы $A_1, A_2, \dots, A_n$ имеют разную размерность и при выполнении операции « $\oplus$ » в случае, если $A_{n-1} > A_n$ или $A_{n-1} < A_n$, меньшая битовая последовательность дополняется до большей битовой последовательности собственным повторением символов, справа, начиная с первого. Выполнение операции « $\oplus$ » осуществляется с сохранением длины записи полученных битовых векторов X_i , т.е. полученные впереди стоящие нули не отбрасываются.

Полученный префикс для битовых векторов $A_1, A_2, \dots, A_n$, полученный по описанному выше алгоритму, дополняется справа непосредственной записью битового вектора A_1 и сохраняется, например, у пользователя. А записанные последовательно друг за другом битовые векторы $X_1, X_2, \dots, X_{n-1}$ передаются на сервер ЦОД.

Также, для получения битовых векторов $X_1, X_2, \dots, X_{n-1}$, вместо операции « $\oplus$ », возможно использовать один из режимов шифрования блочных шифров, таких как CBC (Cipher Block Chaining), CFB (режим гаммирования с обратной связью, Cipher Feedback) [16]. Это необходимо, чтобы убрать возможную корреляцию между битовыми векторами $X_1, X_2, \dots, X_{n-1}$.

Например, пусть, как и предыдущем примере, даны $n = 3$ битовых векторов $A_1 = 111011111101101$, $A_2 = 1010111011101100101$, $A_3 = 10100001010111101101011111010101011$, тогда согласно описанному выше алгоритму получается следующий префикс: 001100001111000001001100000100101111101111101101

Тогда:

$$\begin{aligned} X_1 &= 1010111011101100101101011101110110010 \otimes \\ &\otimes 1010000101011110110101111101010101011 = \\ &= 0000111110110010011000100000100011001 \\ X_2 &= 1010000101011110110101111101010101011 \otimes \\ &\otimes 11101111101101111011111011011110111 = \\ &= 0100111010000101000010000110001011100 \end{aligned}$$

И они записываются последовательно друг за другом 00001111101100100110001000001000110010100111010000101000010000110001011100

Таким образом, если атакующему станет известна последовательность, состоящая из последовательно записанных векторов X_i , он не сможет восстановить исходную последовательность без знания сформированного префикса. Если осуществлять хранение битовой последовательности, например на сервере ЦОД, а префикс у клиента, то будет обеспечиваться должный уровень конфиденциальности хранимой информации.

В случае, если атакующий попытается восстановить исходное изображение фрейма, зная исходные размеры изображения и только битовую последовательность без префикса, хранящуюся на сервере ЦОД, то в результате получит, например, следующее изображение (рис. 4).

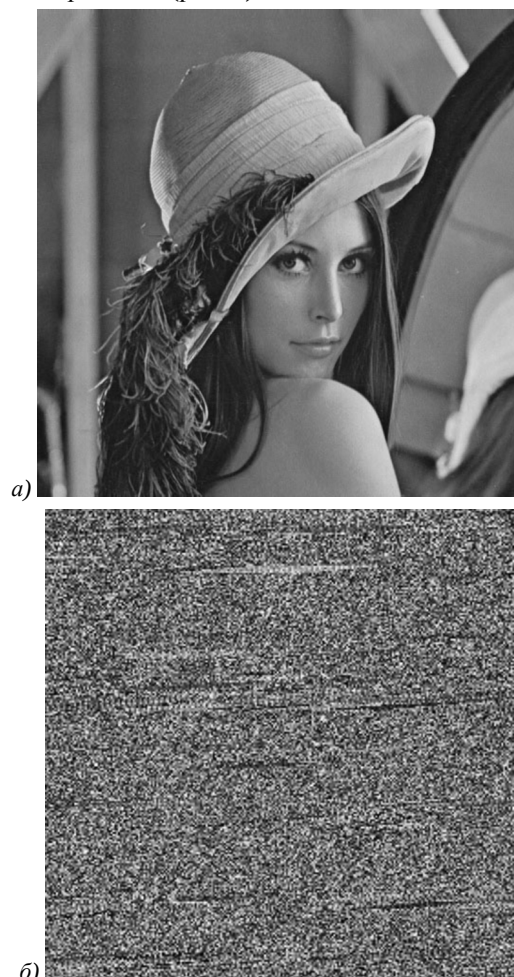


Рис. 4. Исходное изображение фрейма (а), изображение фрейма при попытке восстановления без знания префикса (б)

Аналогичная картина наблюдается и для чёрно-белого изображения с ярко выраженными границами (рис. 5).

Следует отметить, что для одного и того же фрейма при каждом новом преобразовании будет формироваться различный префикс ввиду того, что каждый раз осуществляется разбиение исходной битовой последовательности на произвольные части.

Как видно из полученных изображений (рис. 4б, рис. 5б), даже зная полученную последовательность и применяя всякого рода перестановки, атакующий всё равно не сможет восстановить исходные изображения без знания размеров битовых векторов, полученных в результате разбиения на произвольное число частей исходной битовой последовательности.

Таким образом, для каждого фрейма потока видеoinформации формируется свой префикс – уникальная «ключевая последовательность», что позво-

ляет говорить о преобразовании с использованием изменяющейся ключевой последовательности.

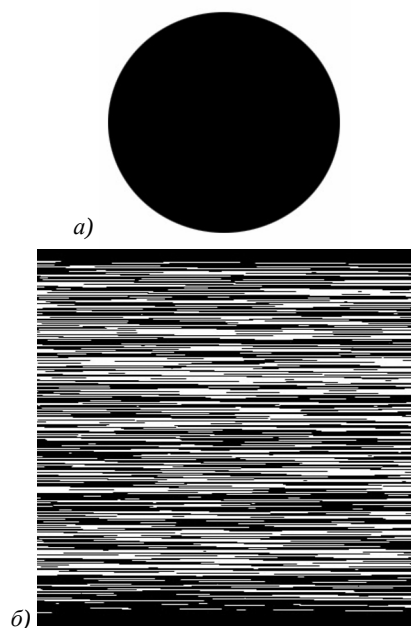


Рис. 5. Исходное изображение фрейма (а), изображение фрейма при попытке восстановления без знания префикса (б)

При наличии у пользователя информации о размере изображения (размер «стандартный» и известен всем), сформированного префикса и битовой последовательности ему удастся восстановить исходное изображение. Также следует отметить, что получаемый префикс для каждого фрейма значительно меньше, чем вся исходная битовая последовательность соответствующего фрейма. Например, для изображения фрейма размером 352×288 , приведённого на рис. 4, вся исходная последовательность равна 2433024 битам и которая произвольно разбита на 98 частей (понятно, что каждый раз получается разное количество частей) от 20000 до 30000 бит, а полученный префикс, который хранится у пользователя, равен 41867 бит. Тогда битовая последовательность, хранящаяся на сервере ЦОД и состоящая из последовательно записанных друг за другом векторов A_i , которые получились, например, после операции « $\oplus$ », равна 3836656 битам, что приблизительно в 91 раз больше полученного префикса.

Но даже если атакующему станет известна часть префикса и данные будут разделены на равные части, но не A_1 , ему всё равно не удастся восстановить исходное изображения, т.к. задача сводится к решению неопределённой системы из n уравнений с $n+1$ неизвестными:

$$\begin{cases} A_2 \otimes A_3 = X_1 \\ \dots\dots\dots \\ A_n \otimes A_1 = X_{n-1}. \end{cases}$$

Восстановить изображение можно только подбо-
ром бит, но в случае, когда длина записи A_1 больше,

чем 80 бит, задача становится принципиально не решаемой, т.к. осуществить перебор на имеющейся в данное время вычислительной технике невозможно.

Восстановление исходного изображения

Пусть пользователь имеет полный префикс, сформированный на стадии разделения секрета, с записанным следом битовым вектором A_1 . А на сервере ЦОД хранится часть из последовательно записанных друг за другом битовых векторов X_i . Размер изображения, стандартный и он известен всем.

1. Вначале осуществляется анализ префикса, имеющегося у пользователя. Анализируется количество первых l бит префикса пользователя, равных нулю. После записи нулей выбирается последовательность, состоящая из нулей и единиц, равная по длине последовательности этих нулей. Выбранная последовательность переводится из двоичной системы счисления в десятичную, и тем самым определяется количество битовых A_i .
2. Следующим этапом подсчитывается количество нулей, записанных после двоичной записи числа битовых векторов. После этих нулей отсчитывается битовая последовательность, равная по количеству подсчитанных нулей. Полученная битовая последовательность равна двоичной записи размера M_1 вектора A_1 .
3. Далее осуществляется операция, аналогичная операции 2, для всего количества векторов A_i , определенного в 1.
4. В результате выполнения пунктов 1 - 3 определяется количество битовых векторов A_i и их длина. Далее из префикса изымается вектор A_1 .
5. Далее с учётом длин битовых векторов A_i решается система уравнений:

$$\begin{cases} A_2 \otimes A_3 = X_1 \\ \dots\dots\dots \\ A_n \otimes A_1 = X_{n-1}. \end{cases}$$

Тем самым находятся все битовые векторы A_i .

Более подробно рассмотрим восстановление на примере. Пусть имеется префикс, полученный в приведённом выше примере: 0011000011110000010 0110000001001011101111101101 и некоторая битовая последовательность 000011111011001001100 01000001000110010100111010000101000010000110 001011100.

Вначале проанализируем префикс. Получается, что имеется три битовых вектора A_i , длина M_1 вектора A_1 равна 15, длина M_2 вектора A_2 равна 19, длина M_3 вектора A_3 равна 37 и вектор $A_1 = 11101111101101$.

С учётом того, что длина M_3 вектора A_3 равна 37 и длина M_i вектора A_i равна 15, отсчитываем с конца битовой последовательности 37 знаков и находим вектор A_3 . $A_3 = 010011101000010100001000011000 1011100 \otimes 11101111101101111011111011011110111 =$

= 1010000101011110110101111010101011. Далее, с учётом того, что длина M_2 вектора A_2 равна 19 и длина M_3 вектора A_3 равна 37, отсчитываем справа налево следующие 37 знаков битовой последовательности и находим вектор A_2'

$A_2' = 0000111110110010011000100000100011001 \otimes$
 $\otimes 101000010101111011010111101010101011 =$
 $= 101011101110110010110101110110110010.$

Зная, что длина M_2 вектора A_2 равна 19, от полученной последовательности A_2' отсчитываем слева направо 19 символов и получаем, что вектор $A_2 = 1010111011101100101$. Тем самым найдены все три вектора A_1, A_2, A_3 .

Заключение

Описанные методы не требуют значительных вычислительных ресурсов и способны осуществлять преобразование данных «на лету». Следует отметить, что в случаях, если атакующему стал известен предыдущий кадр потока видеoinформации, то не исключены попытки построения алгоритмов вскрытия, основанных на корреляционной зависимости, что требует дальнейшего исследования.

Для демонстрации каждого из предложенных методов были разработаны программы, реализующие описанные выше действия. Средой разработки является Borland C++ Builder 6.

Благодарность

Работа выполнена при финансовой поддержке РФФИ (грант № 12-07-00294-а).

Литература

1. RFC-2205, -2209, -2210, -1990, -1889, -3550, -3551, -3989, -3952; «RTP: A Transport Protocol for Real-Time Applications», H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. URL: <http://tools.ietf.org/html/rfc3550>.
2. RFC 2326 «Real Time Streaming Protocol (RTSP)» H. Schulzrinne, A. Rao, R. Lanphier. URL: <http://www.ietf.org/rfc/rfc2326.txt>.
3. RFC 3261 «SIP: Session Initiation Protocol». J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler. URL: <http://www.ietf.org/rfc/rfc3261.txt>.
4. RFC 4123 «Session Initiation Protocol (SIP)-H.323 Interworking Requirements». H. Schulzrinne, C. Agboh. URL: <http://tools.ietf.org/html/rfc4123>.
5. Файзуллин, Р.Т. Алгоритм разделения секрета с использованием принципиально малой части секрета в качестве ключа / Р.Т. Файзуллин, И.Р. Файзуллин, О.Т. Данилова // Вестник Тюменского государственного университета. – 2011. – № 7. – С. 175-179.
6. Шнайер, Б. Прикладная криптография: протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер. 2-е издание. – М.: Триумф, 2002. – 610 с.
7. Lintian, Q. Comparison of MPEG Encryption Algorithms / Q. Lintian, K. Nahrstedt // International Journal on Computers and Graphics, Special Issue: Data Security in Image Communication and Network. – 1998. – Vol. 22.

8. Shamir, A. How to share a secret / A. Shamir // Communications of the ACM 22. – 1979. – P. 612-613.
9. Чин-Пан, Х. Совместное использование секретных изображений с применением обычных вейвлетов и мультивейвлетов / Хуанг Чин-Пан // Радиотехника и электроника. – 2011. – Т. 56, № 1. – С. 53-62.
10. Свенч, А.А. Схема разделения секрета на основе метрических характеристик данных для защищенной передачи видеоконференций / А.А. Свенч, Р.Т. Файзуллин // Компьютерная оптика. – 2007. – Т. 31, № 1. – С. 105-111.
11. Файзуллин, Р.Т. Построение системы защиты видеоданных с использованием решения задач проверки изоморфизма графов / Р.Т. Файзуллин, Г.С. Ржаницын // Компьютерная оптика. – 2006. – № 29 – С. 84-93.
12. Rahul, R. (July 28, 2008). Shard – A Database Design [Электронный ресурс] / Roy Rahul. – Режим доступа: <http://technoroy.blogspot.com/2008/07/shard-database-design.html>, свободный (дата обращения 29.04.2012).
13. Файзуллин, И.Р. Аппаратно-эффективный алгоритм кодирования и шифрования данных, основанный на специальной задаче выбора / И.Р. Файзуллин, Р.Т. Файзуллин // Омский научный вестник. Радиотехника и связь. – 2010. – № 3(93). – С. 248-250.
14. Elias, P. Universal codeword sets and representations of the integers / P. Elias // IEEE Transactions on Information Theory. – 1975. – Vol. IT-21, N 2. – P. 194-203.
15. Federal Information Processing Standards Publication. FIPS PUB 140-1. Security Requirements for Cryptographic Modules. – U.S. Department of commerce / National institute of standards and technology, 1994. – 53 p.
16. Recommendation for Block Cipher Modes of Operation. NIST Special Publication 800-38A. Technology Administration U.S. Department of Commerce. 2001 Edition.

References

1. RFC-2205, -2209, -2210, -1990, -1889, -3550, -3551, -3989, -3952; «RTP: A Transport Protocol for Real-Time Applications», H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. URL: <http://tools.ietf.org/html/rfc3550>.
2. RFC 2326 «Real Time Streaming Protocol (RTSP)» H. Schulzrinne, A. Rao, R. Lanphier. URL: <http://www.ietf.org/rfc/rfc2326.txt>.
3. RFC 3261 «SIP: Session Initiation Protocol». J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler. URL: <http://www.ietf.org/rfc/rfc3261.txt>.
4. RFC 4123 «Session Initiation Protocol (SIP)-H.323 Interworking Requirements». H. Schulzrinne, C. Agboh. URL: <http://tools.ietf.org/html/rfc4123>.
5. Faizullin, R.T. Secret sharing algorithm with a fundamentally small part of a secret as the key / R.T. Faizullin, I.R. Faizullin, O.T. Danilova // Bulletin of the Tyumen State University. – 2011. – N 7. – P. 175-179. – (In Russian).
6. Schneier, B. Applied Cryptography, 2nd Edition: protocols, algorithms, source code in C / B. Schneier. – Moscow: «Triumph» Publisher, 2002. – 610 p. – (In Russian).
7. Lintian, Q. Comparison of MPEG Encryption Algorithms / Q. Lintian, K. Nahrstedt // International Journal on Computers and Graphics, Special Issue: Data Security in Image Communication and Network. – 1998. – Vol. 22.
8. Shamir, A. How to share a secret / A. Shamir // Communications of the ACM 22. – 1979. – P. 612-613.
9. Chin-Pan, H. Sharing the secret-dimensional images using conventional wavelets and multiwavelets / Huang Chin-Pan

- // Technology and Electronics. – 2011. – Vol. 56, N 1. – P. 53-62. – (In Russian).
10. **Svench, A.A.** Secret sharing scheme based on the metric characteristics of data for secure transmission of video conferencing. / A.A. Svench, R.T. Faizullin // Computer Optics. – 2007. – V. 31, N 1. – P. 105-111. – (In Russian).
 11. **Faizullin, R.T.** Building security system with video verification tasks graph isomorphism / R.T. Faizullin, G.S. Rzhantsyn // Computer Optics. – 2006. – N 29. – P. 84-93. – (In Russian).
 12. **Rahul, R.** (July 28, 2008). Shard – A Database Design / Roy Rahul. – <http://technoroy.blogspot.com/2008/07/shard-database-design.html>, free (date accessed 29/04/2012).
 13. **Faizullin, I.R.** Hardware-efficient encoding algorithm and data encryption, based on the special problem of the choice / I.R. Faizullin, R.T. Faizullin. // Omsk Scientific Gazette. Radio engineering and communication. – 2010. – N 3(93). – P. 248-250. – (In Russian).
 14. **Elias, P.** Universal codeword sets and representations of the integers / P. Elias // IEEE Transactions on Information Theory. – 1975. – Vol. IT-21, N 2. – P. 194-203.
 15. Federal Information Processing Standards Publication. FIPS PUB 140-1. Security Requirements for Cryptographic Modules. – U.S. Department of commerce / National institute of standards and technology, 1994. – 53 p.
 16. Recommendation for Block Cipher Modes of Operation. NIST Special Publication 800-38A. Technology Administration U.S. Department of Commerce. 2001 Edition.

MODEL SECRET SHARING SCHEMES IN SYSTEMS TRANSMIT VIDEO

*D.A. Sagaydak, R.T. Faizullin
Omsk State Technical University*

Abstract

This paper proposes two embodiments secret sharing schemes on non-equal shares with essentially a small part of the secret to securely storage and processing of video data centers, which are based on RGB-representation (abbreviation round English words Red, Blue, Green - Red, green, blue) of each video frame. For each of the proposed scheme is made software implementation in the programming environment Borland C++ Builder 6.

Key words: data processing center (DPC), secret sharing, video sharing, representation in RGB, frame, delta code Elias, pixel representation of the image, prefix, distrust of the data center, the mafia attack.

Сведения об авторах



Сагайдак Дмитрий Анатольевич, аспирант Омского государственного технического университета кафедры комплексной защиты информации. Область научных интересов: криптография, компьютерное моделирование, математические расчёты, программирование.

E-mail: sagaydak.dmitriy@gmail.com.

Dmitriy Anatolyevich Sagaydak, a graduate school student at Omsk State Technical University Department «Complex Protection of Information» Research interests: cryptography, computer simulations, mathematical calculations, programming.



Файзуллин Рашид Тагирович, доктор технических наук (1999 г.), профессор Омского государственного технического университета кафедры комплексной защиты информации. В списке научных работ Р.Т. Файзуллина более 100 статей, 2 монографии.

E-mail: frt@omgtu.ru.

Rashit Tagirovich Faizullin, Doctor of Technical Sciences (1999), Professor of Omsk State Technical University Department «Complex Protection of Information». He is co-author of more when 100 scientific papers, 2 monographs.

Поступила в редакцию 4 октября 2012 г.