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 защищенного хранения и обработки видеоинформации в центрах обработки данных, в основе которых лежит RGB-представление (аббревиатура английский 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 позволяют осуществлять потоковую передачу данных, зачастую, такими данными является медиаконтент (аудио-, видеоданные). Для передачи потоковой видеоинформации используются технологии сжатия и буферизации данных (двойная, тройная, z-буферизация), позволяющие передавать видеоинформацию в режиме реального времени с помощью протоколов потокового вещания, например, таких как: RTP, RTSP, RTCP, SIP, H.323 и другие [1,2,3,4].

Повышение качества видеоизображения, возобновление интереса к стереооптическому видео (трехмерное видео), появление видео высокого разрешения/четкости (HDTV) - все это способствует увеличению объемов медиа-контента. В настоящее время для хранения объемного медиа-контента все 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 так же снижает расходы на их техническую поддержку. Но по-прежнему остается актуальным вопрос обеспечения конфиденциальности передаваемой на хранение в ЦОД информации. Даже поддержанные лицензиями уровни надежности и меры обеспечения защиты информации в ЦОД не могут убедить клиентов в их надежности, так угроза мафиозной атаки [6] очень высока. Ввиду этого, клиенты ЦОД, в качестве которых могут выступать, как обычные пользователи, так и крупные организации, размещающие свой медиа-контент, в целях достижения конфиденциальности передаваемых видеоданных, осуществляют самостоятельное шифрование этих данных. Но опять же, выбор алгоритма шифрования возлагается на клиентов ЦОД, которые могут не являться специалистами по защите информации и не могут обеспечить должный уровень защиты. Таким образом, получается противоречие. Владелец видеоинформации предполагает хранить ее 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.

Так же 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 опять же, 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. Опять же основным недостатком данной схемы являются высокие требования к производительности оборудования в связи с необходимостью осуществлять шифрование больших объемов 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. А так же, критична по требованию к производительности оборудования и сторона, на которой осуществляется дешифрование принимаемых видеоданных (клиентская сторона ЦОД). Часто на практике применяются, например, следующие методы защиты видеоданных в формате семейства MPEG: метод случайной перестановки ко-

эффициентов дискетного косинусного преобразования (ДКП), метод селективного шифрования, метод шифрования на основе изменяемых кодовых таблиц, которые опять так же 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. В виду того, что технические возможности ЦОД позволяют хранить большие объемы 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 осуществляется принципиального акцента на размер части секрета. А как так же таковые отсутствуют работы по приложению схем разделения секрета для обеспечения конфиденциальности видеоданных, когда те хранятся на серверах ЦОД. Тем самым в данной работе будут предложены два варианта схем разделения секрета на неравные доли, где часть секрета будет являться лишь малой частью передаваемой видеоинформации.

Описание первой схемы разделения секрета

Пусть имеется некий несжатый поток видеоинформации расширением *MxN*. Первоначально осуществляется деление данного потока видеоинформации 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-х 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-х битовой последовательности.

Над каждой такой последовательностью выполняется следующая перестановка:

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

Далее последовательность $\overline{T}$ 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 создается массив A размерностью $6 \cdot M$, состоящий из полученных α_i для каждого пикселя пиксельной строки изображения.

Далее находится среднее значение массива A:

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

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

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

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

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

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

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

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

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 $\overline{F}$ размерностью $6 \cdot M \cdot N$ и 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 в данном случае открытой информацией является массив $\overline{F}$ размерностью $6 \cdot M \cdot N$. Пользователь получит ключевую последовательность, состоящая из изъятых знаков и мест их изъятия.

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

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

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

Из рис.1*б* 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 два раунда. Полученное изображение после первого раунда поворачивается на 90 градусов (по часовой или против часовой стрелки) и осуществляется второй раунд. Так же возможно, например, 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 $\overline{F}$, состоящего из элементов $|f_i|$, после двух раундов преобразования исходного фрейма без осуществления какой-либо перестановки столбцов и строк изображения после первого раунда преобразования.

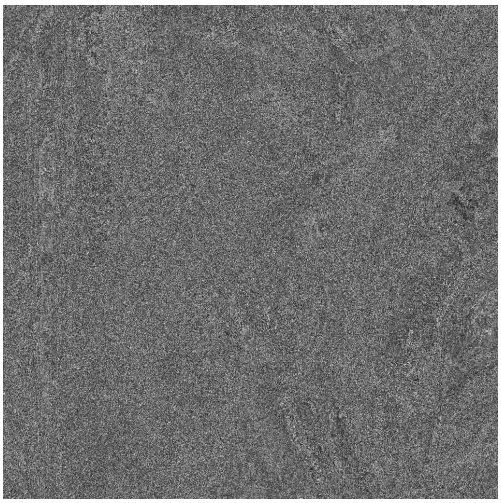

Рис.2 Изображение фрейма при попытке восстановления по элементами массива $\overline{F}$ после двух раундов преобразования исходного изображения фрейма

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

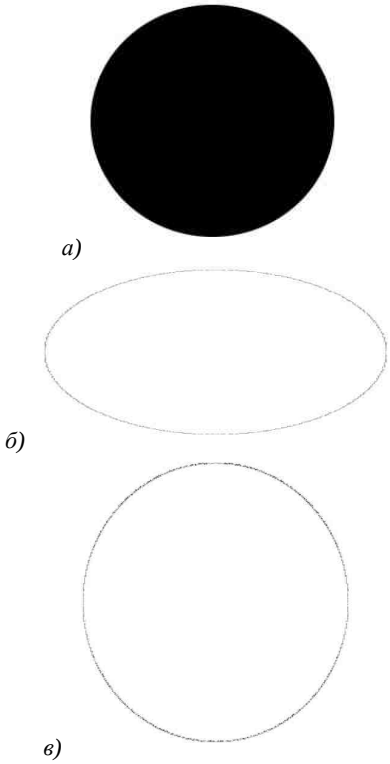

Рис.3 Исходное изображение фрейм (а), изображение фрейма после первого раунда преобразования (б), изображение фрейма после двух раундов преобразования (в). Фон изображений (б) и (в) инвертирован (вместо белого фона подразумевается чёрный фон)

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, приведенной в (2). Для данной матрицы находятся собственные числа λ_i и собственные векторы υ_i, тогда секрет можно восстановить по следующему принципу: $F\overline{A} = S$, где $\overline{A}$ – искомый секрет.

1. $S = \sum_{i=1}^{6\cdot M} \beta_i \cdot \upsilon_i \Rightarrow \beta_i = (S, \upsilon_i)$, т.е. выполняется скалярное произведение векторов S и υ_i, где S - ключ, представляющий из себя числовой вектор, β_i - скалярное произведение вектора S, на собственные векторы υ_i матрицы H.

2. Находим элементы вектора $\overline{A} = \sum_{i=1}^{6\cdot M} \frac{\beta_i}{\lambda_i} \cdot \upsilon_i$

Элементы исходного массива A равны $a_i = sr_A - \overline{a_i}$. Выполнив обратную перестановку, перестановке описанной выше, т.е. $\overline{T} \Rightarrow T$, и осуществив преобразование значение 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 $A_1,...,A_n$, размерности которых равны $M_1,...,M_n$ 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, например, такими как: шардинг [12], т.е. грубым физическим разделением 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 битовых 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 $m_1 = 000$, $m_2 = 000$, $m_3 = 00$. Тем самым, 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:
00110001100001010011111011101111101.

Обратим внимание на то, что, зная диапазон возможных изменений A_i, можно записывать A_i в $m_i + d_i$ позициях, предваряя 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 видеоинформации, и мы попытаемся построить алгоритм, не требующий значительных вычислительных ресурсов.

Пусть имеется поток определенного («телевизионного») формата 720x576 пикселей 25 кадров в секунду в формате RGB (в дальнейшем будет осуществляться преобразование видеопотоков стандартных форматов: 720x576, 640x480, 352x288 (CIF – Common Interchenge Format), 176x144 (QCIF – Quartered Common Interchenge 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 $A_1, A_2, ..., A_n$, разных размерностей $M_1,...,M_n$.

В качестве примера, для генерации размерностей $M_1,...,M_n$. битовых векторов $A_1, A_2, ..., A_n$ 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 удовлетворяет общим требованиям к датчикам псевдослучайных чисел, описанных 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 $A_1, A_2, ..., A_n$ осуществляются следующие операции: $X_1 = A_2 \otimes A_3$, $X_2 = A_3 \oplus A_4, \ldots,$ $X_{n-2} = A_{n-1} \oplus A_n$, $X_{n-1} = A_n \oplus A_1$ (где «$\oplus$» - побитовое сложение по модулю 2), полученные битовые векторы $X_1, X_2, ..., X_{n-1}$ записываются последовательно друг за другом. Следует заметить, что битовые векторы $A_1, A_2, ..., A_n$ 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 впередистоящие нули не отбрасываются.

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

Так же, для получения битовых векторов $X_1, X_2, ..., X_{n-1}$, 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 $X_1, X_2, ..., X_{n-1}$.

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: 0011000011110000010011000001010111101111101101
Тогда:
$X_1 = 10101110111011001011010111011101100 10 \otimes$
$\otimes 10100001010111101101011111010101011 =$
$= 00001111101100100110001000000100011001$
$X_2 = 10100001010111101101011111010101011 \otimes$
$\otimes 11101111110110111101111110111011110111 =$
$= 01001110100001010000100001100010111 00$
И они записываются последовательно друг за другом 00001111101100100110001000001000110010100
11101000010100001000011000101110 0

Тем самым, 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).

а)

б)

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

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, в виду того, что каждый раз осуществляется разбиение исходной битовой последовательности на произвольные части.

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

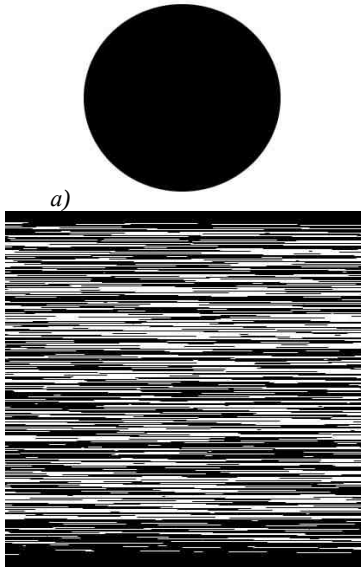

Рис.5 Исходное изображение фрейма (а), изображение фрейма при попытке восстановления без знания префикса

Таким образом, понятно, что для каждого фрейма потока видеоинформации формируется свой уникальный префикс – уникальная «ключевая последовательность», что позволяет говорить о том, что осуществляется преобразование с использованием изменяющейся ключевой последовательностью.

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 352x288, приведённого на рис.4., вся исходная последовательность равна 2433024 бита, 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 бит, что в 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 все равно не удастся восстановить исходное изображения, т.к. задача сводится к решению неопределенной системы из n уравнений с $n+1$ неизвестными:

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

Восстановить изображение можно только подбором 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. Потом после записи нулей выбирается последовательность, состоящая из нулей и единиц равная по длине последовательности этих нулей, и 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 и 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 аналогичная операция операции 2) для всего количества векторов A_i, определенного в 1).
4) Тем самым 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 учетом длин битовых векторов A_i решается система уравнений:

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

Тем самым находятся все битовые векторы A_i.

Более подробно рассмотрим восстановление на примере. Пусть имеется префикс, полученный в приведенном выше примере: 0011000011110000010 01100000010010111101111111011 01 и некоторая битовая последовательность 000011111011001001100 01000001000110010100111010000101000010000110 001011100 .

Вначале проанализируем префикс, получатся 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 $A_1 = 111011111101101$

С учетом того, что длина M_3 вектора A_3 равна 37 и длина M_1 вектора A_1 равна 15, отсчитываем с конца битовой последовательности 37 знаков и находим вектор A_3. $A_3 = 0100111010000101000010000011000$ $1011100 \otimes 11101111110110111101111110110111101111 =$ $= 10100001010111101101011111010101011011$. Далее с учетом того, что длина M_2 вектора A_2 равна 19 и длина M_3 вектора A_3 равна 37, отсчитываем справа на лево следующие 37 знаков битовой последовательности и находим вектор $A_2^{'}$

$A_2^{'} = 0000111110110010011000100000100011001 \otimes$
$\otimes 10100001010111101101011111010101011 =$
$= 10101110111011001011010111011101100010$

Зная, что длина M_2 вектора A_2 равна 19 то от полученной последовательности $A_2^{'}$ отсчитываем слева на право 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 требует значительных вычислительных ресурсов и способны осуществлять преобразование данных «на лету». Следует отметить, что в случаях, если атакующему стал известен предыдущий кадр потока видеоинформации, то не исключены попытки построения алгоритмов вскрытия основанные 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., Файзуллин И.Р., Данилова О.Т.** Алгоритм разделения секрета с использованием принципиально малой части секрета в качестве ключа. // Вестник Тюменского государственного университета, № 7 2011, С. 175-179.
6. **Шнайер Б.** Прикладная криптография, 2-е издание: протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер. - М.: Триумф, 2002. - 610 с.
7. **Lintian Qiao** and **Klara Nahrstedt**. Comparison of MPEG Ecryption Algorithms // International Journal on Com-puters and Graphics, Special Issue: Data Security in Image Communication and Network, 1998, vol. 22.
8. **Shamir A.** How to share a secret // Communications of the ACM 22, 1979. С. 612-613.
9. **Чин-Пан Хуанг.** Совместное использование секретных изображений с применением обычных вейвлетов и мультивейвлетов. // Радиотехника и электроника, Т. 56б, № 1, С. 53-62.
10. **Свенч А.А., Файзуллин Р.Т**. Схема разделения секрета на основе метрических характеристик данных для защищенной передачи видеоконференций. // Компьютерная оптика, Т.31, №1, 2007, Самара ИСОИ РАН, - С. 105-111.
11. **Файзуллин Р.Т., Ржаницын Г.С.** Построение системы защиты видеоданных с использованием решения задач проверки изоморфизма графов. // Компьютерная оптика, № 29 2006, ИСОИ РАН, Самара, ИСОИ РАН – С. 84-93.
12. **Rahul Roy** (July 28, 2008). Shard - A Database Design [Электронный ресурс]. – Режим доступа: http://technoroy.blogspot.com/2008/07/shard-database-design.html, свободный (дата обращения 29.04.2012).
13. **Файзуллин И.Р., Файзуллин Р.Т.** Аппаратно-эффективный алгоритм кодирования и шифрования данных, основанный на специальной задаче выбора. // Омский научный вестник. Радиотехника и связь, №3 (93), 2010, Омск, – С. 248-250.
14. **Elias P.** Universal codeword sets and representations of the integers // IEEE Transactions on Information Theory, 1975, vol. IT-21, № 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 с.
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. **R.T. Faizullin, I.R. Faizullin, O.T. Danilova.** Secret sharing algorithm with a fundamentally small part of a secret as the key. // Bulletin of the Tyumen State University, № 7, 2011, P. 175-179.
6. **B. Schneier,** Applied Cryptography, 2nd Edition: protocols, algorithms, source code in C / B. Schneier. - M: Triumph, 2002. - 610 p.

7. **Lintian Qiao** and **Klara Nahrstedt**. Comparison of MPEG Ecryption Algorithms // International Journal on Com-puters and Graphics, Special Issue: Data Security in Image Communication and Network, 1998, vol. 22.
8. **Shamir A.** How to share a secret // Communications of the ACM 22, 1979. P. 612-613.
9. **Chin-Pan Huang**. Sharing the secret-dimensional images using conventional wavelets and multiveyvletov. / / Technology and Electronics, Vol 56b, № 1, P. 53-62.
10. **A.A. Svench, R.T. Faizullin**. Secret sharing scheme based on the metric characteristics of data for secure transmission of video conferencing. // Computer Optics V.31, № 1, 2007, Samara IPSI RAS - P. 105-111.
11. **R.T. Faizullin, G.S. Rzhanitsyn**. Building security system with video verification tasks graph isomorphism. / / Computer Optics, № 29, 2006, IPSI RAS, Samara, IPSI RAS - P. 84-93.
12. **Rahul Roy** (July 28, 2008). Shard - A Database Design [electronic resource]. - Mode of access: http://technoroy.blogspot.com/2008/07/shard-database-design.html, free (date accessed 29/04/2012).
13. **I.R. Faizullin, R.T. Faizullin.** Hardware-efficient encoding algorithm and data encryption, based on the special problem of the choice. // Omsk Scientific Gazette. Radio engineering and communication, № 3 (93), 2010, Omsk - P. 248-250.
14. **Elias P.** Universal codeword sets and representations of the integers // IEEE Transactions on Information Theory, 1975, vol. IT-21, № 2. P. 194-203.
15. Federal Information Processing Standards Publication. FIPS PUB 140-1. Security Requirements for Cryptographic Modules. - US 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.

Сагайдак Дмитрий Анатольевич, аспирант Омского государственного технического университета кафедры «Комплексная защита информации» Область научных интересов: криптография, компьютерное моделирование, математические расчёты, программирование.

Эл. почта: 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 Рашит Тагирович, доктор технических наук (1999 г.), профессор Омского государственного технического университета кафедры «Комплексная защита информации». В списке научных работ Р.Т. Файзуллина более 100 статей, 2 монографии.

Эл. почта: 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.