ЭССЕ по курсу «Защита информации»

Алгоритм RC4 и его применение в стандарте 802.11 беспроводных сетей

Студента 915 группы Слынько Ю.В.

Введение

В сентябре 1999 IEEE-SA одобрило два стандарта беспроводных сетей: 802.11b - 2.4 ГГц, 11 Мб и 802.11a - 5 ГГц, 54 Мб. С тех пор беспроводные сети быстро распространились во все ниши рынка, такие как торговля, образование, здравоохранение и т.д. Важным фактором в повсеместном распространении беспроводных сетей и включении их в существующие сети является потребность вездесущего доступа к основным бизнес приложениям и ресурсам 24 часа в сутки, и потребность в установке базовых сервисов и персональных данных вдали от конечного пользователя. В дополнении, возможность взаимодействия сетей была облегчена независимыми тестовыми организациями, такими как Wireless Ethernet Compatibility Alliance (WECA).

Это действительно правда, что мобильность – это основное требование в локальных сетях предприятий. Предполагается, что к 2003 будет более миллиарда мобильных устройств.

Однако, некоторые препятствия, такие как безопасность, высокая доступность, распределение спектра и т.д. у первого поколение беспроводных сетей мешают предприятиям развёртывать крупномасштабные беспроводные сети. В данном эссе рассмотрен вопрос безопасности беспроводных сетей, который для предприятий является основным препятствием, ограничивающим применение беспроводных сетей только там, где проводные просто нежизнеспособны.

802.11 – первое поколение WLAN безопасности

Между проводными и беспроводными сетями существует несколько различий. Самое важное состоит в том, что здесь нет проводов (вместо них – «воздушные связи») и мобильность есть их неотъемлемая часть. Поскольку беспроводная связь не ограничена проводом, здесь действительно утверждение, что данные в беспроводных сетях, рассылающиеся широковещательно, не защищены. Проводные сети должны быть физически вскрыты для того, чтобы можно было прослушать информацию, распространяющуюся по ним. Беспроводные сети наоборот, могут быть прослушаны кем угодно с подходящей антенной.

Для решения этой проблемы в стандарте 802.11 содержатся механизмы защиты MAC-уровня для защиты содержимого кадров, распространяющихся по сети, от несанкционированного доступа. В первом поколении беспроводных сетей было две области, связанные с безопасностью:

- SSID (Service Set Identifier) – установленный сервисом идентификатор
- WEP (Wired Equivalent Privacy) – эквивалентная проводной секретность

В дополнении к этим способам существует другом распространённый способ усиления безопасности – использование виртуальных частных сетей (VPN), которые проходят непосредственно через беспроводные сети. Но в данном эссе этот способ не рассматривается, т.к. он не зависит от стандарта.

SSID

Чаще всего используемая в беспроводных сетях вещь – это именной идентификатор, SSID, который обеспечивает рудиментарный уровень защиты.

SSID – это аналог обыкновенному имени сети для беспроводных станций и точек доступа в данной беспроводной подсистеме. SSID служит логическому сегменту пользователей и точек доступа, которые образуют часть беспроводной подсистемы. SSID – это кусочек информации, который может быть отконфигурирован вручную на станции или объявлен сервером. SSID может быть запрошен зондирующим запросом (Probe-request frame), когда узел пытается присоединиться к беспроводной подсети, или объявлен как часть периодического сигнального кадра, посылаемого точкой доступа.

В любом случае, использование SSID как способа разрешения или запрещения доступа довольно опасно, т.к. обычно он недостаточно защищён. Фактически, для того, чтобы точки доступа работали в 802.11 в гибком режиме, обычно устанавливается широковещательные режим рассылки SSID (Broadcast- SSID mode), другими словами, они получают SSID в их сигнальных кадрах (beacons). Несмотря на это, довольно много беспроводных сетей не используют ничего, кроме SSID, для разрешения доступа для неавторизованных пользователей.

WEP

IEEE 802.11b стандарт пытается обеспечить «защищенность провода» посредством не обязательной схемы шифрование, названной WEP (Wired Equivalent Privacy). WEP, несмотря не необязательность, доступен как механизм первого поколения обеспечения защищённого взаимодействия между узлами и защиты потоков данных в беспроводных сетях. Члены WECA неизменно поддерживают как минимум 40-битное шифрование как часть демонстрации меж узлового взаимодействия. Основные цели WEP – это:

- Запретить доступ к сети неавторизованным пользователям, которые не обладают подходящим WEP ключом.
- Предотвратить дешифрование захваченного трафика, который WEP зашифровал, без знания WEP ключа.

WEP – это механизм симметричного шифрования. Если WEP разрешён, передатчик берёт содержимое кадра, т.е. полезную нагрузку, и запускает алгоритм шифрование на нём. Затем оригинальное содержимое кадра заменяется на выход алгоритма шифрования. Кадры данных, которые были зашифрованы, посылаются с WEP битом в контрольном поле MAC заголовка. Получатель кадра с зашифрованными данными пропускает кадр через тот же алгоритм шифрования, что и посылающая сторона. В результате получается оригинальный кадр, который может быть передан протоколам высшего уровня. Другими словами WEP – это симметричная схема шифрования.

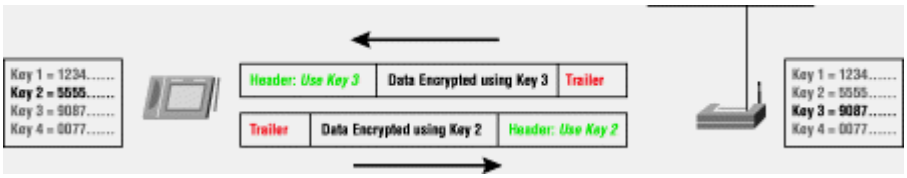
Производительность WEP зависит от вида реализации – аппаратной или программной, а также от конкретного устройства. Некоторые устройства позволяют добиться производительности, лишь на 2-3 процента худшей, чем без использования шифрования. Однако зачастую, особенно при программной реализации, происходит существенное уменьшение производительности сети.

WEP использует потоковый шифр RC4, который был изобретён Роном Ривестом из RSA Data Security, Inc. (RSADSI). Алгоритм шифрования RC4 – это симметричный потоковый шифр, который поддерживает различные длины ключа. Симметричный шифр это шифр, использующий один и тот же ключ для зашифровывания и расшифровывания. Он сильно отличается от блоковых шифров, которые обрабатывают фиксированное число байт. Ключ – это некая информация, которая должна быть доступна обеим сторонам – зашифровывающей и расшифровывающей. RC4 допускает различную длину ключа – до

256 байт. В IEEE 802.11b выбрана длина ключа в 40 бит. Однако некоторые производители поддерживают также и 128 битный ключ и предоставляют устройства для работы с такой длиной ключа.
Стандарт IEEE 802.11 описывает использование алгоритма RC4 и ключей в беспроводных сетях. Однако распределение ключей или их продажа не упоминается в стандарте. Также, производители сетевой аппаратуры могут выбрать для использования собственные приложения и интерфейсы для поддержки и конфигурирования WEP ключей. К сожалению, эти упущения могут свести на нет все усилия разработчиков данного стандарта. Если производитель оборудования не обеспечил должную защиту ключа, то и информация, передающаяся по беспроводной сети, недостаточно защищена.

Стандарт IEEE 802.11 обеспечивает два механизма выбора ключа для зашифровывания и расшифровывания кадров.

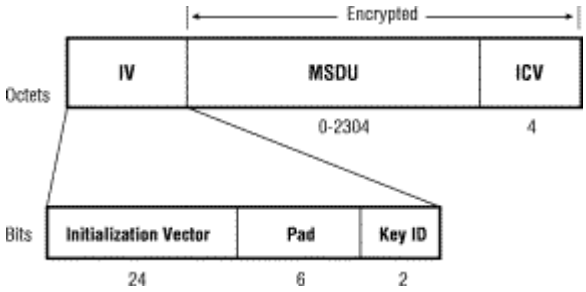
Первый механизм заключается в установке четырёх ключей по умолчанию. Ключи по умолчанию должны быть известны всем станциям беспроводной подсети. Достоинство использования ключей по умолчанию заключается в том, что если станция получила эти ключи, она может общаться секретно со всеми остальными станциями подсети. Недостаток же в том, что т.к. ключи доступны всем станциям, то достаточно велика вероятность их взлома или несанкционированного прочтения неавторизованным пользователем.



Второй механизм, обеспечивающийся стандартом IEEE 802.11, позволяет станции установить взаимодействие с каждой другой станцией по определённым различным ключам («key mapping»). Это, вероятно, более защищённая форма работы, т.к. меньше станций знают ключ. Однако, распределение таких ключей проблематично, если количество станций в сети достаточно велико.

WEP заголовок и концевик добавляется к зашифрованному телу кадра. Номер ключа по умолчанию, который нужно использовать для расшифровывания кадра содержится в поле KeyID заголовка кадра вместе с инициализационным вектором. Концевик содержит Integrity Check Value (ICV) для контроля правильности переданного кадра.

Длина ключа обычно делится на длину WEP ключа и длину инициализационного вектора. Например, 64 битный ключ состоит из 40 битного WEP ключа, хранящегося секретно, и 24 битного инициализационного вектора.



IEEE 802.1 аутентификация и соединение

Аутентификация – это процесс проверки удостоверения личности клиента, пытающегося подсоединится к сети. Соединение – это процесс соединения клиента с данной точкой доступа к беспроводной сети.

В стандарте 802.11, в действительности, существует три состояния пользователя:

1. Неавторизованный и не присоединённый
2. Авторизованный и не присоединённый
3. Авторизованный и присоединённый

IEEE 802.11 определяет два типа аутентификационых методов: открытая система аутентификации и аутентификация с разделённым (shared) ключом. Удачное выполнение фаз аутентификации и соединения позволяет узлу беспроводной сети удачно войти в беспроводную подсеть.

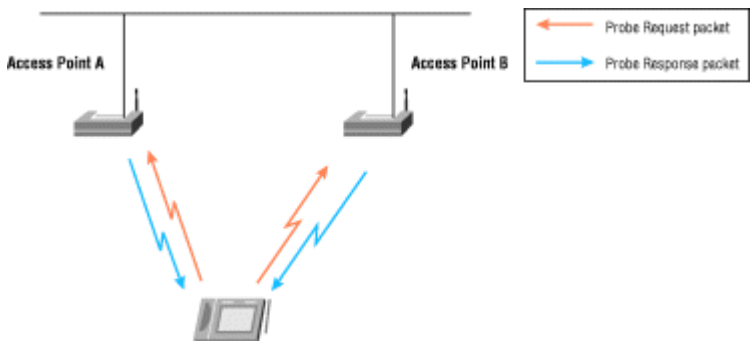
При аутентификации с открытым ключом весь аутентификационный процесс проходит открытым текстом. Это значит (т.к. весь процесс проходит незашифрованным) что клиент может присоединиться к точке доступа с неправильным WEP ключом или вообще без WEP ключа. Но, как только клиент попытается послать или принять данные, он не сможет этого сделать, т.к. для обработки кадров необходимо знать правильный ключ. При аутентификации с разделённым ключом в процессе аутентификации используются зашифрованные сообщения. Если клиент не обладает верным ключом, то он не пройдёт стадию аутентификации и не сможет присоединиться к точке доступа.

Выбор между методами аутентификации производится вручную на каждом устройстве, но методы клиента и сервера должны совпадать, чтобы аутентификация прошла успешно. По умолчанию используется открытая аутентификация.

Весь процесс может быть разделён на три фазы:

Фаза зондирования (Probe)

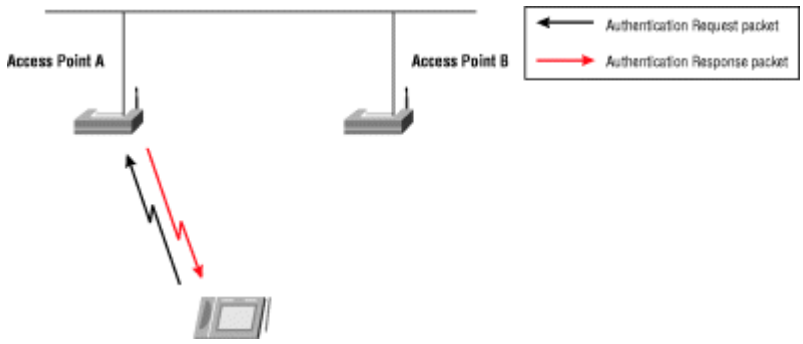
Когда клиент инициализируется, он сначала посылает зондирующий запрос по всем каналам. Точки доступа, которые слышат этот запрос, посылают зондирующий ответ назад на станцию. Зондирующий ответ содержит такую информацию, как SSID, который клиент хранит для определения, какая точка доступа продолжает процесс соединения.



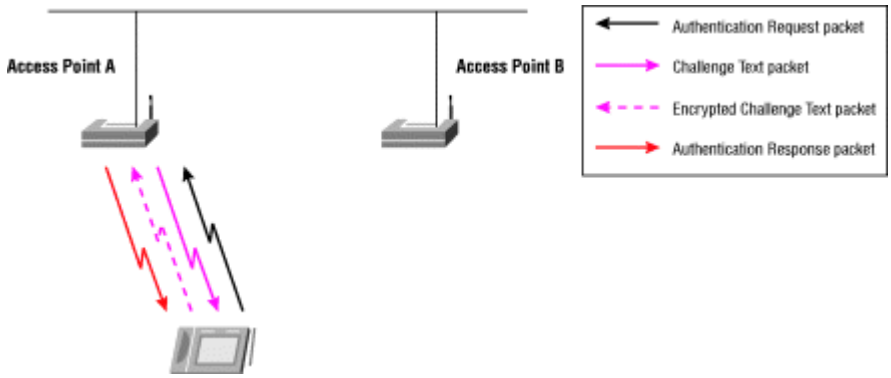
Фаза аутентификации

После того, как клиент определят, какая точка доступа продолжает процесс соединения, он начинает процесс аутентификации на основе зондирующего ответа. Эта фаза может быть выполнена как в открытом режиме, так и в режиме разделённого ключа. Обе стороны – сервер и клиент – должны быть настроены на одинаковый режим аутентификации, иначе аутентификация не пройдёт.

Схема открытой аутентификации:
Клиент посылает аутентификационный запрос на точку доступа. Точка доступа обрабатывает этот запрос и определяет (основываясь на собственной конфигурации), позволить или не позволить клиенту перейти к фазе соединения. Точка доступа посылает аутентификационный ответ назад клиенту. Основываясь на типе ответа (аутентификация прошла или нет) от точки доступа, клиент продолжает или прекращает процесс соединения.



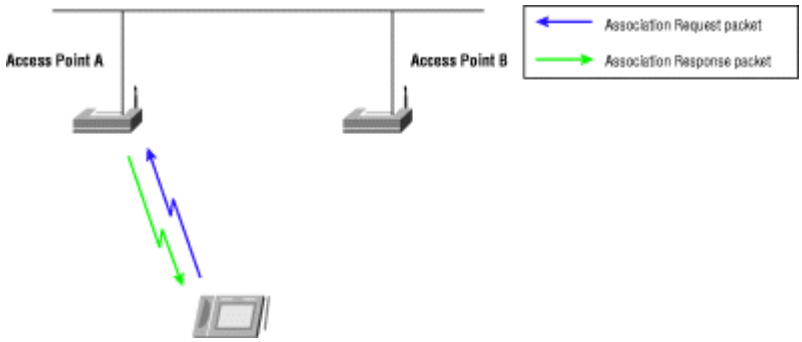
Аутентификация с разделённым ключом:
Клиент посылает аутентификационный запрос на точку доступа. Точка доступа обрабатывает этот запрос, генерирует и посылает зашифрованный текст клиенту. Клиент затем должен расшифровать сообщение с помощью секретного WEP ключа и послать пакет назад на точку доступа. Точка доступа определяет, удалось ли клиенту верно расшифровать пакет. Основываясь на этом тесте, точка доступа посылает положительный или отрицательный аутентификационный ответ клиенту, который определяет, позволено ли клиент продолжать процесс соединения или нет.



Фаза соединения

Если клиент удачно прошёл фазу аутентификации (например, получил положительный аутентификационный ответ от точки доступа), он начинает фазу соединения. Клиент посылает запрос на соединение точке доступа. Точка доступа анализирует информацию в

этом запросе и если она правильная, точка доступа добавляет клиента в свою таблицу соединений. Затем она посылает ответ клиенту, и фаза соединения на этом завершается.



Недостатки стандарта 802.11

Самый существенный недостаток стандарта 802.11 заключается в отсутствии хорошего способа распределения ключей. Используемое статическое распределение практически не применимо в чистом виде для больших сетей и на больших промежутках времени. Возникает необходимость периодического изменения ключей, а для сетей больших масштабов, и при отсутствии удобных способов распределения ключей, это составляет большую проблему для системного администратора. Также особенно трудно сохранить ключ секретным, если к сети должны иметь доступ неавторизованные пользователи (скажем, если одна из станций предоставляется в аренду).

Также статическое распределение ключей чревато тем, что при пассивном наблюдении трафика сети достаточно долго, возможно накопить достаточно информации о ключе, что позволит беспрепятственно дешифровать сообщения.

Устранение этого недостатка требует создание схемы динамического распределения ключей, их постоянного обновления и привязки ключей не к узлу сете, а к пользователю (чтобы неавторизованный пользователь знал только свой личный ключ, и в случае его широкого распространения третьей стороне стал доступен только его трафик).

Второй существенный недостаток состоит в том, что в стандарте 802.11 существует только процесс аутентификации клиента, а аутентификация сервера не происходит. Это даёт возможность атаковать сеть путём введения в неё несанкционированных серверов и перенаправления потока данных на эти сервера.

Для устранения этого недостатка необходимо ввести схему взаимной аутентификации, в которой обе стороны должны доказывать свою легитимность. Если же узел не смог этого сделать за разумное время, это должен быть изолирован по подозрению в несанкционированности.

Следующим недостатком является то, что не происходит аутентификация каждого пакета, и возможны подмены пакетов или генерирование несуществующих. Хотя это довольно трудно сделать технически, хакеры делают практически невозможное для взлома сети, и случаи атака таким способом имели место.

Для устранения этого недостатка необходимо чаще сменять ключи и инициализационный вектор. Также обсуждается вопрос введения стандарта шифрования AES, взломать который значительно труднее, чем RC4.

Также, в некоторых реализация WEP ключи извлекают из паролей, фраз, SSID’ов и т.д. Тогда возможно их угадывание либо вообще по детерминированному алгоритму, либо перебором по словарю.

Однако от этого недостатка легко избавиться, если отказаться от системы вычисления ключей, а выбирать их случайным образом.

Также можно производить такие атаки, как, например, посылать серверам сообщения о разрыве чужого соединения. Т.к. эти сообщения передаются незашифрованными, то вероятность успешной атаки очень велика. Однако особого урона в плане безопасности это не наносит.

Алгоритм RC4

В стандарте 802.11 используется алгоритм шифрования RC4.

RC4 - это потоковый шифр. Он широко применяется в различных системах защиты информации в компьютерных сетях.

Основные преимущества шифра - высокая скорость работы и переменный размер ключа. Типичная реализация выполняет 19 машинных команд на каждый байт текста. В США длина ключа для использования внутри страны рекомендуется равной 128 битов, но соглашение, заключенное между Software Publishers Association (SPA) и правительством США дает RC4 специальный статус, который означает, что разрешено экспортировать шифры длинной ключа до 40 бит. 56-битные ключи разрешено использовать заграничным отделениям американских компаний.

Шифр разработан в RSA, его автор - Ronald Rivest. RC расшифровывается как " Ron's Code" или " Rivest Cipher". Шифр является коммерческим секретом RSA, до 1995 г. Не публиковался, но различным разработчикам предоставлялась лицензия на его использование.

Алгоритм RC4 строится, как и любой потоковый шифр, на основе параметризованного ключом генератора псевдослучайных битов с равномерным распределением.

Надежность RC4 была недавно поставлена под сомнение из-за несанкционированного разглашения текста программы, использующей этот шифр.

В 1995 некто опубликовал анонимно в телеконференции sci.crypt исходный текст алгоритма RC4. По-видимому, данный текст был получен в результате анализа исполняемого кода. Опубликованный шифр совместим с имеющимися продуктами, использующими RC4, а некоторые участники телеконференции, имевшие, по их словам, доступ к исходному коду RC4, подтвердили идентичность алгоритмов при различиях в обозначениях и структуре программы.

Алгоритм RC4 состоит из трех частей:

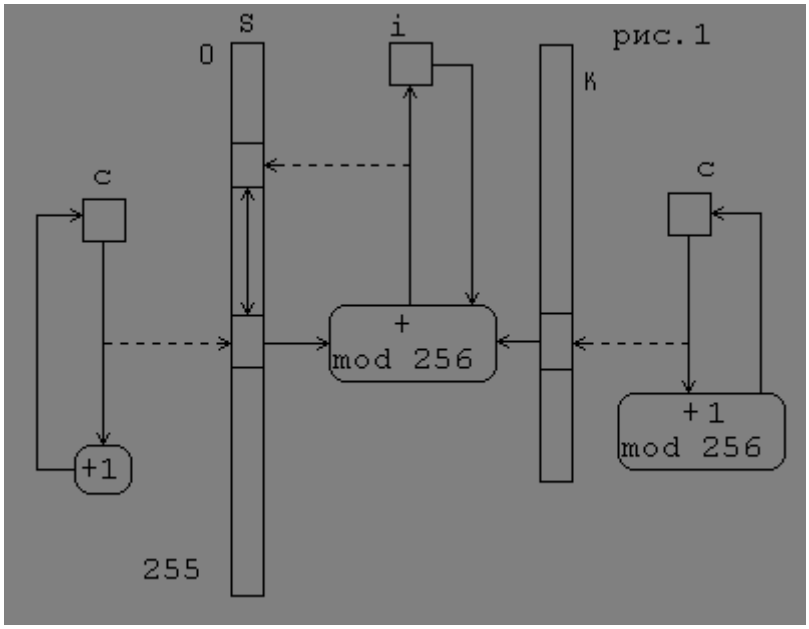
1. Создание ключа (иногда называют - расширение ключа).
2. Алгоритм шифрования.

Создание ключа

Ключ в RC4 представляет собой последовательность байтов произвольной длины, по которой строится начальное состояние шифра ***S*** - перестановка всех 256 байтов. Алгоритм получения начального состояния изображен на рис.1.

Первоначально ***S*** заполняется последовательными значениями от 0...255, а ***K*** заполняется ключом (при необходимости для заполнения всего массива ключ повторяется). Затем каждый очередной элемент S_c обменивается местами с элементом с номером ***i***, номер которого определяется элементом ключа ***K***, самим элементом и суммой номеров элементов, с которыми происходил обмен на предыдущих итерациях, т.е. $i=S_c+K_c+i$.

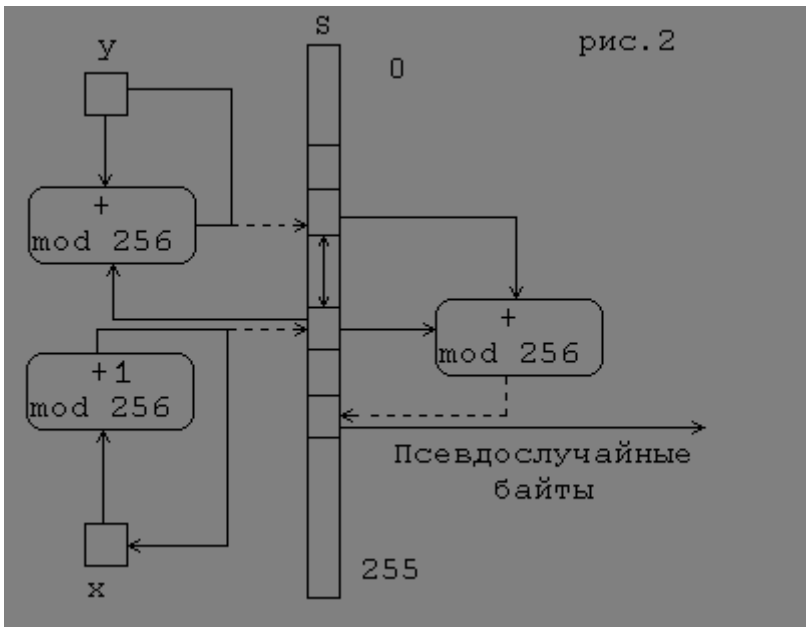
Значения счетчиков ***i*** и ***c*** изначально равны 0. Сплошные стрелки означают передачу значений между элементами схемы (присваивание), двусторонние стрелки - обмен значениями, пунктирные стрелки - индексацию в массиве.



Алгоритм шифрования

Собственно алгоритм псевдослучайных битов в RC4 схематически изображен на рис. 2.

Очередной элемент псевдослучайной перестановки S_x всех байтов обменивается с элементом S_y, где $x=x+1 \bmod 256$, а $y=y+S_x \bmod 256$. В качестве очередного байта выдается значение третьего элемента ***S***, номер которого равен сумме S_x и S_y. Значение счетчика ***x*** первоначально равно 0, но оно увеличивается на 1 уже перед первой выборкой S_x. Значение ***y*** также первоначально равно 0.



Текст алгоритма на C

```
#define buf_size 1024
#define swap_bate(x,y)  t = *(x); *(x) = *(y); *(y) = t
typedef struct rc4_key //ключ
   {
        unsigned char state[256];
        unsigned char x;
        unsigned char y;
   } rc4_key;
  //подготовка ключа
void prepare_key(usigned char *key_data_ptr, int key_data_len, rc4_key *key)
   {
        unsigned char swapByte;
        unsigned char index1;
        unsigned char index2;
        unsigned char* state;
        short counter;

        state = &key->state[0];
        for(counter = 0; counter < 256; counter++)
            // S заполняется последовательными значениями от 0...255
            state[counter] = counter;
        key->x = 0;
        key->y = 0;
        index1 = 0;
        index2 = 0;

        for(counter = 0; counter < 256; counter++)
        {    // каждый очередной элемент S обменивается местами с элементом
             index2 = (key_data_ptr[index1] + state[counter] + index2) % 256;
             //номер которого определяется элементом ключа key
             //самим элементом и суммой номеров элементов, с которыми
происходил об
             //мен на предыдущих итерациях.
             swap_byte(&state[counter], &state[index2]);
             index1 = (index1 + 1) % key_data_len;
        }
    }
//алгоритм шифрования RC4
```

```
    void rc4(unsigned char *buffer_ptr, int buffer_len, rc4_key *key)
    {
        unsigned char x;
        unsigned char y;
        unsigned char* state;
        unsigned char xorIndex;
        short counter;

        x = key->x;
        y = key->y;
        state = &key->state[0];

        for(counter = 0; counter < buffer_len; counter ++)
        {
             x = (x + 1) % 256;
             y = (state[x] + y) % 256;
             swap_byte(&state[x], &state[y]);
             //Очередной элемент псевдослучайной перестановки  state всех
байтов обменивается с
            //другим, номер которого равен сумме элементов, выбрнных на
предыдущих шагах.
             xorIndex = (state[x] + state[y]) % 256;
              // В качестве очередного байта выдается значение третьего
элемента state,  номер
              //   которого равен сумме первых двух.
             buffer_ptr[counter] ^= state[xorIndex];
         }

         key->x = x;
         key->y = y;
    }

//Главная программа
int main (int argc, char *argv[])
 {
   char seed[256];
   char data[512];
   char buf[buf_size];
   char digit[5];
   int hex, rd, i, n;
   rc4_key key;

   clrscr();
   if (argc < 2) { fprint(stderr, "%s key out\n", argv[0]; exit(1);}
   strcopy(data,argv[1]);
   n = strlen(data);
   if (n&1) { strcat(data,"0"); n++}
   n/=2;
   strcpy(digit, "AA");
   digit[4] = '\x0';
   for (i = 0; i < n; i++)
      {
        digit[2] = data[i*2];
        digit[3] = data [i*2+1];
        sscanf(digit, "%x",&hex);
        seed[i] = hex;
      }
   prepare_key(seed, n, &key);
   rd = fread(buf, 1, buf_size, stdin);
   while (rd >0)
     {
       rc4(buf, rd, &key);
       fwrite(buf, 1, rd, stduot);
       rd = fread(buf, 1, buf_size, stdin);
```

```
    }
}//end programm
```

Заключение

Основной проблемой беспроводных сетей является их низкая безопасность и незащищённость от несанкционированного доступа, если они применяются «в чистом виде», без специальных систем безопасности.

Стандарт беспроводных сетей 802.11 обеспечивает достаточно высокий уровень безопасности и надёжности, при этом почти не снижая быстродействие сети, что обеспечивает широкое распространение беспроводных сетей по всему миру во всех областях деятельности.

Однако этот стандарт не лишен ряда существенных недостатков, и сейчас уже ведётся разработка стандарта безопасности второго поколения.

Использованная литература и WEB ресурсы

- Брюс Шнайдер «Прикладная криптография»
- www.citforum.ru/internet/securities
- www.ciscosystems.am/warp/public/102/wlan