Уязвимости WEP

Выполнил	студент
Институт	МФТИ
Факультет	РТК
Курс	4
Группа	211
ФИО	Михин Дмитрий Юрьевич
Документ	эссе по курсу «Защита информации»

Уязвимости WEP

Название документа	Автор	Дата	Ревизия	Описание
Уязвимости WEP	Дмитрий Михин	2006.02.20	0.1	Начальный набросок
Уязвимости WEP	Дмитрий Михин	2006.03.08	0.2	Написанны разделы: 1.2.1, 1.2.2, 1.2.3
Уязвимости WEP	Дмитрий Михин	2006.03.09	0.3	Написанны разделы: 1.2.4.1, 1.2.4.2
Уязвимости WEP	Дмитрий Михин	2006.03.11	0.4	Написанны разделы:1.2.4.3, 1.2.4.4, 2
Уязвимости WEP	Дмитрий Михин	2006.03.12	1.0 RC1	Исправление неточностей, замеченных друзьями.
Уязвимости WEP	Дмитрий Михин	2006.03.25	1.0	Исправлен 1.2.3, написан 1.3

1. WEP

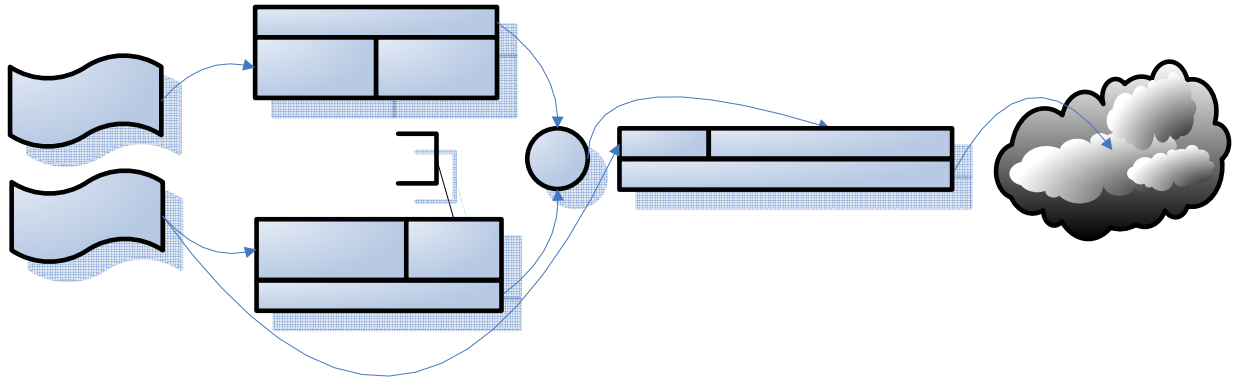
Протокол 802.11 подразумевает под собой сетевой обмен между множеством устройств по одной беспроводной среде. Оборудование, находящееся в определённой зоне «видимости» может получать данные от других участников обмена информации. Но зачастую данные предназначены не для всех устройств, находящихся в зоне «видимости», а только для зарегистрированных пользователей. Поэтому консорциумом IEEE был предусмотрен алгоритм, который должен был бы выполнять следующие функции:

- Конфиденциальность. Данные, передаваемые по беспроводной среде, могут быть прочитаны только зарегистрированными пользователями
- Контроль доступа. Неавторизированные устройства не имеют доступ к сети и её ресурсам
- Целостность данных. Невозможность осуществления модификации передаваемых данных

Название WEP - Wired Equivalent Privacy, означает, что он должен предоставлять уровень защиты, адекватный в сравнении с проводной сетью.

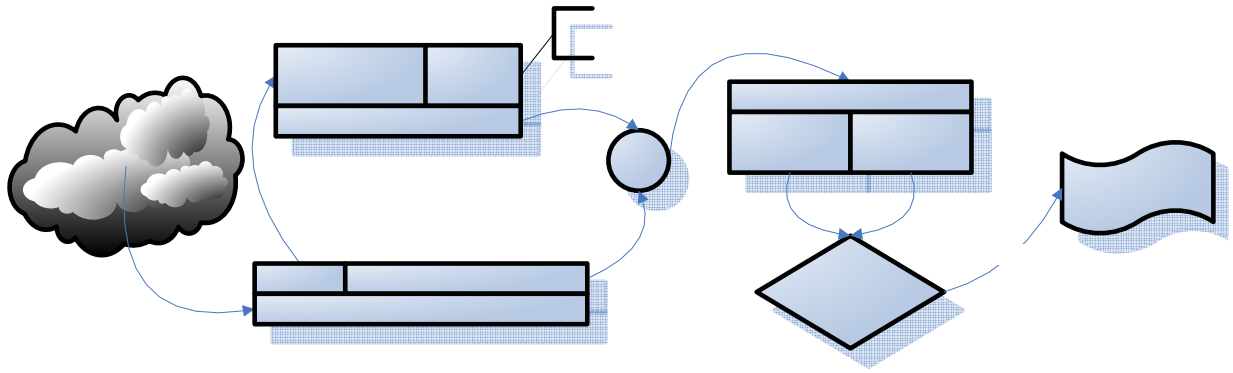
1.1 Описание алгоритма

Всем зарегистрированным пользователям известен секретный ключ K, о котором они заранее договариваются. Для остальных устройств, находящихся в зоне «видимости» этот ключ неизвестен. Процесс шифрования происходит следующим образом:



- Для сообщения M высчитывается контрольная сумма по алгоритму CRC32 [5], $ICV = CRC(M)$ и прибавляется к самому сообщению: $P = \{M, ICV\}$
- Выбирается вектор инициализации IV (алгоритм его выбора не оговорен) и, зная секретный ключ K, генерируется ключевой поток по алгоритму RC4 [4]: $PRS = RC4(IV, K)$
- Складываем ключевой поток PRS и суммарное сообщение P (с контрольной суммой), используя исключающее ИЛИ – XOR, получается шифротекст $C = P \oplus PRS$.
- Шифротекст вместе с инициализирующим вектором образуют целое – «пакет» $EP = \{IV, C\}$, он передаётся по беспроводной среде.

Расшифрование происходит следующим образом:



- Из полученного «пакета» EP берётся вектор инициализации IV и, зная секретный ключ K, генерируется ключевая последовательность $PRS = RC4(IV, K)$.
- Ключевая последовательность складывается с шифротекстом C, в результате получается сообщение с контрольной суммой: $C \oplus PRS = P \oplus PRS \oplus PRS = P = \{M, ICV\}$
- Выполняется проверка $CRC(M) = ICV$. Если она верна, то операция расшифрования прошла успешно, M - переданное сообщение.

1.2 Возможные атаки

Мы не будем рассматривать алгоритм создания ключевой последовательности PRS, нам важно лишь знать, что для одних и тех же IV и K она будет одна и та же. Будем считать, что длины пакетов одинаковы (это упростит выкладки, и не изменит общего смысла). Длина вектора инициализации в стандартной реализации - 24 бита, а длина ключа K - 40 бит. Ключ K - фиксированный, задаётся при создании сети. Фактически шифрование и расшифрование происходит только при помощи ключевого потока PRS, и знание его для некоторых IV равносильно знанию ключа K для этих IV.

1.2.1 Повторное использование ключевого потока

Так как K не меняется, то PRS зависит только от IV. Сам IV может принимать $2^{24} \approx 16 \cdot 10^6$ значений. То есть после передачи $16 \cdot 10^6$ пакетов, вектор инициализации обязательно повторится. Время, через которое это произойдёт, при работе в загруженной сети на скорости 11Мбод (802.11b), принимая длину пакета равной 1500 байт, будет $\frac{16 \cdot 10^6 \cdot 1500 \cdot 8}{11 \cdot 10^6} \approx 18000$ секунд или 5 часов. Если же IV выбираются случайным образом, то 50%-ый шанс получения коллизии возможен уже после передачи 4823-его пакета. Рассмотрим два пакета, зашифрованные при одном и том же IV:

$$C_1 = P_1 \oplus PRS$$
$$C_2 = P_2 \oplus PRS$$
$$PRS = RC4(IV, K)$$
$$C_1 \oplus C_2 = P_1 \oplus PRS \oplus P_2 \oplus PRS = P_1 \oplus P_2$$

То есть мы легко можем получить сумму незашифрованных сообщений. Т.к. предаваемые данные в сети стандартизированы (содержат определённые заголовки, контрольные суммы), из суммы сообщений зачастую легко получить каждое сообщение P_1 и P_2 в отдельности.

Далее, дешифровав один пакет, описанным выше способом, или любым другим (сообщение можно и не получать дешифрованием, а просто угадать, стандартные ARP, RARP, DHCP, ICMP запросы), вычисляем $PRS = C \oplus P$ и заносим в словарь пару $[IV, PRS]$. В следующий раз, когда перехватим пакет с таким IV, просто применим операцию XOR с PRS и получим сообщение.

1.2.2 Внедрение и подмена пакетов

Знание пары $[IV, PRS]$ даёт возможность не только дешифровать сообщение с таким IV, но и посылать сообщения. Злоумышленник хочет послать сообщение M, тогда он вычисляет $C = \{M, ICV = CRC(M)\} \oplus PRS$, и затем передаёт пакет $\{IV, C\}$. С точки зрения получателя – это валидный пакет, т.к. правильно зашифрован, следовательно, сообщение будет принято.

Также передаваемый пакет можно изменить, связано это с линейностью алгоритма CRC32 [5]: $CRC(M_1 \oplus M_2) = CRC(M_1) \oplus CRC(M_2)$. D - некоторое значение, которое выбрал злоумышленник, так, чтобы навязать получателю изменённый пакет $M \oplus D$. Криптоаналитик перехватил пакет $C = \{M, ICV = CRC(M)\} \oplus PRS$, затем он выполняет следующую операцию: $C_{new} = C \oplus \{D, CRC(D)\}$ и передаёт получателю $\{IV, C_{new}\}$. Тогда:

$$C_{new} = \{M, CRC(M)\} \oplus \{D, CRC(D)\} \oplus PRS = \{M \oplus D, CRC(M) \oplus CRC(D)\} \oplus PRS = \\ = \{M \oplus D, CRC(M \oplus D)\} \oplus PRS$$

С точки зрения получателя – это валидный пакет, т.к. правильно зашифрован, следовательно, сообщение $M \oplus D$ будет принято.

1.2.3 Аутентификация незарегистрированного пользователя

Самая курьёзная ошибка в дизайне WEP, это, пожалуй, Shared Key аутентификация. Она работает следующим образом:

- Базовая станция (точка доступа) отсылает клиенту квитанцию (challenge string) открытым текстом
- Клиент шифрует квитанцию на ключе K и передаёт базовой станции (обычное WEP шифрование)
- Базовая станция проверяет правильность шифрования, используя имеющийся у неё ключ K, и, в случае успеха, аутентифицирует клиента

То есть мы автоматом получаем пару $[IV, PRS]$. А далее, мы можем выдавать себя за легитимного клиента:

- Базовая станция посылает нам (challenge string) CS
- Отвечаем пакетом $\{IV, CS \oplus PRS\}$
- Это корректный ответ (response string), и базовая станция принимает нас, хотя мы никогда не знали ключ K

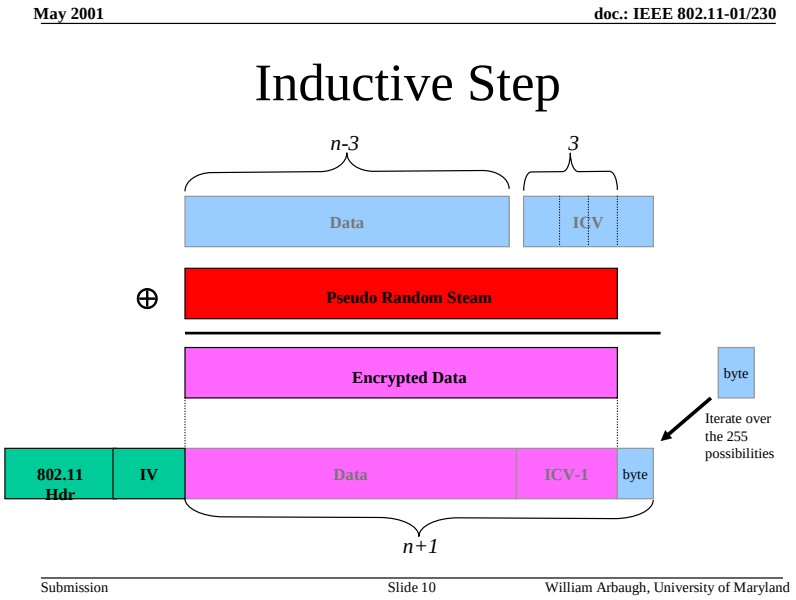
Поэтому во многих продуктах отказались от Shared Key аутентификации, которая только компрометирует сеть, вместо неё используется Open System аутентификация. При использовании открытой аутентификации никакой аутентификации, собственно, и не существует, то есть любой пользователь может получить доступ в беспроводную сеть.

1.2.4 Дешифрование пакетов

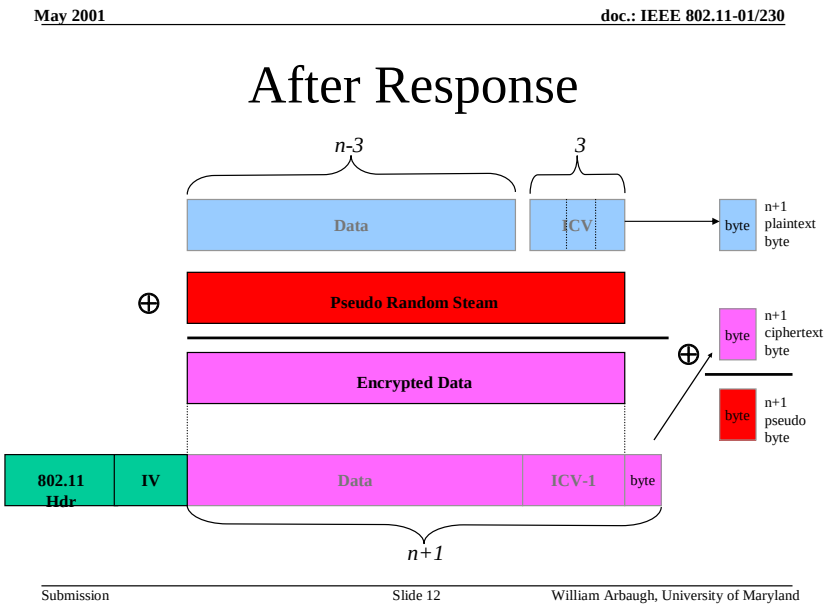
В данном разделе будут рассмотрены более продвинутые атаки. Здесь так же будет учтено, что пакеты имеют разную длину, и соответственно ключевую последовательность PRS разной длины.

1.2.4.1 Получение всей ключевой последовательности (Arbaugh inductive attack [7])

Допустим, нам уже известна часть ключевой последовательности PRS, мы хотим узнать оставшуюся её часть. Для этого достаточно проделать следующее:



- Создаём сообщение, размера $n-3$, где n - количество известных байтов из PRS
- Считаем для него ICV (4ёх байтная контрольная сумма CRC32 [5]), и припишем к сообщению только 3 байта.
- Сложим с известными n байтами PRS
- Припишем к сумме сзади произвольный $(n+1)$-ый байт
- Добавляем спереди IV (также заголовок 802.11) и посылаем сообщение точке доступа

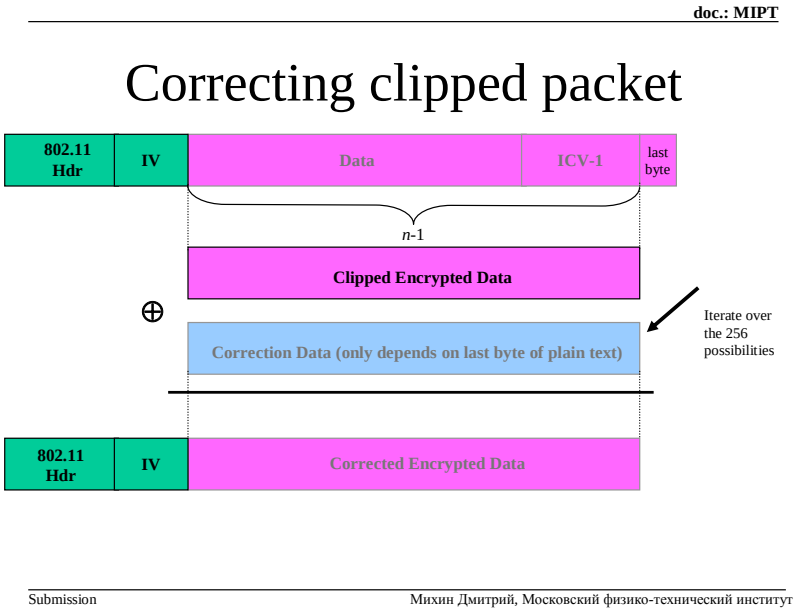


Если приписанный $n+1$ байт оказался таким, что после расшифровки, выполняется проверка $ICV = CRC(Data)$, то точка доступа пошлёт Ack-ответ (подтверждение о получении), если злоумышленник был аутентифицирован в сети или deauth-ответ, если злоумышленник не был аутентифицирован в сети, если же проверка не выполняется, станция просто отбросит этот пакет, не совершая никаких ответных действий. Перебирая все 256 возможных значений $n+1$ байта, мы обязательно получим ответ. После того как ответ получен, легко вычисляется $n+1$ байт ключевой

последовательности, как сумма 4ого байта посчитанного ранее ICV и предположенного значения $(n+1)$-ого байта.

1.2.4.2 Дешифрование без знания ключевой последовательности (chopchop [11])

Особенностью данной атаки является, то, что мы сравнительно легко можем сразу получить и ключевую последовательность, и сам текст зашифрованного сообщения. Последовательность действий следующая:



- Отрезаем от зашифрованного сообщения 1 байт
- Теперь чтобы оно стало валидным, сообщение нужно поправить (изменять сам пакет мы уже умеем, см. 1.2.2). Для этого высчитываем специальный «вектор исправления», предположив, что последний байт незашифрованного сообщения равен b (как будет показано дальше, вектор зависит только от b)
- Складываем «вектор исправления» с «обрезанными» зашифрованными данными
- Добавляем спереди IV (также заголовок 802.11) и посылаем сообщение точке доступа

Если предположение о последнем байте незашифрованного сообщения b было сделано верно, то мы получим в ответ Ack или deauth-ответ, если же предположение было неверным, в качестве b берём другое значение. Так последовательно, отрезая последний байт у зашифрованного сообщения, исправляя его, и посылая в сеть, можно восстановить всё сообщение, а, следовательно, и ключевую последовательность PRS.

Теперь покажем, что «вектор исправления» действительно зависит только от последнего байта, и покажем, как этот вектор можно получить. Для этого вспомним, как работает алгоритм CRC. Чтобы понять, что будет написано, нужно иметь представление о полях Галуа [6]. Итак, CRC для сообщения $M(x)$ считается следующим образом $M(x)\cdot x^n = Q(x)\cdot K(x)+R(x)$, где $K(x)$ - многочлен степени n ($x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$ для CRC32), а $R(x)$ - и служит значением CRC32. В результате приписывания ICV к сообщению, получается $M(x)\cdot x^n + R(x) = 0 \bmod K(x)$. Рассмотрим сначала, что будет, если от сообщения отрезать один бит. Чтобы оно снова осталось валидным, нужно найти такое изменение, в результате которого мы опять бы получали $0 \bmod K(x)$. Рассмотрим сообщение, как многочлен: $M(x)\cdot x^n + R(x) = a_k x^k + a_{k-1}x^{k+1} + ... + a_1 x + a_0 = x\left(a_k x^{k-1} + a_{k-1}x^{k-2} + ... + a_1\right) + a_0$, где многочлен в

скобках, как раз соответствует сообщению, с одним отрезанным битом, обозначим его $M_{-1}(x)$. Тогда получим:

$$x \cdot M_{-1}(x) + a_0 = 0 \bmod K(x)$$
$$M_{-1}(x) = a_0 \cdot \frac{1}{x} \bmod K(x)$$

Найдём многочлен, обратный x. $A(x) \cdot x = 1 \bmod K(x) = K(x) + 1 \bmod K(x)$. Т.к. $K(x)$ неприводимый многочлен, то он всегда имеет свободный член 1, тогда $K(x)+1$ - многочлен, который всегда делиться на x. И $A(x) = \frac{K(x)+1}{x}$ ($x^{31} + x^{25} + x^{22} + x^{21} + x^{15} + x^{11} + x^{10} + x^9 + x^7 + x^6 + x^4 + x^3 + x^1 + 1$ для CRC32). Итак $M_{-1}(x) + a_0 \cdot A(x) = 0 \bmod K(x)$. Итак мы нашли многочлен, который нужно прибавить, чтобы сообщение вновь стало валидным, и показали, что оно зависит только от последнего отрезанного бита a_0 (многочлен $A(x)$ - константа, от a_0 не зависит). Повторяя такую операцию ещё 7 раз, мы придём к выводу, что «вектор исправления» зависит только от последнего байта сообщения.

1.2.4.3 Атака с использованием фрагментации (Fragmentation Attack [8])

Одна из самых быстрых атак, для её осуществления нужно, чтобы у беспроводной сети был выход в Интернет. Она основана на том, что при фрагментации пакета, отдельные фрагменты шифруются независимо. Осуществить её можно следующим способом:

- Получить PRS для какого-либо IV, одним из известных способов
- Составить IP-заголовок, который бы направлял интересующий нас пакет, на адрес нашей машины в Интернете (на самом деле нужно сначала добавить Logical-Link Control заголовок, посредством которого в 802.11 могут инкапсулироваться не только IP пакеты, но сути это не меняет)
- Зашифровать IP-заголовок при помощи PRS, и отправить его как 1-ый фрагмент
- Затем взять перехваченное сообщение, и отправить его как 2-ой фрагмент (модифицировать нужно только незашифрованный заголовок 802.11)

В результате точка доступа примет два фрагмента, расшифрует, а затем соединит. Получиться пакет, который содержит интересуемое сообщение, а адресом назначения у него будет наша машина в Интернете. Проблемы могут возникнуть с сообщениями, размер которых больше $MTU - 28$ (MTU – Maximum Transmission Unit, максимальный размер IP-пакета), т.к. полученные соединённое послание будет превышать MTU (20 байт идёт на IP-заголовок плюс 8 байт на Logical-Link Control заголовок). Для таких сообщений можно применить следующее:

- Непосредственно изменить адрес назначения в зашифрованном сообщении. Для этого мы должны знать первоначальный адрес получателя, и нужно будет также изменить контрольную сумму в IP-заголовке, чтобы сообщение осталось валидным. Более подробно это описано Борисовым [9]
- Совершить атаку chopchop, укоротив сообщение на нужное количество байт, чтобы к нему можно было добавить новый заголовок
- Для TCP-сообщений в IP-заголовке выставлен бит Don't Fragment. Если такое сообщение придёт на роутер с более низким MTU, он сам не будет разбивать его на части. Вместо этого он пошлёт ICMP-ответ «message too big». TCP-отправитель понизит Maximum Segment Size, до значения MTU роутера, и после этого перешлёт сообщение более мелкими кусками. Мы можем использовать такую же технику, что бы заставить отправителя понизить размер TCP сообщения.

1.2.4.4 Статистические атаки

Они основаны на слабости алгоритма RC4, когда часть битов входного ключа оказывает значительное влияние на выходной поток. Впервые публично об этой проблеме написали Scott

Fluhrer, Itsik Mantin и Adi Shamir в статье «Weaknesses in the Key Scheduling Algorithm of RC4». Теперь все атаки подобного типа называются FMS (по первым буквам авторов). Более детальное рассмотрение выходит за рамки данного эссе.

1.3 Улучшения WEP

Осознав опасность, при использовании WEP, сами производители оборудования попытались улучшить его, введя поддержку 128-битных ключей (24 бит IV и оставшиеся 104 бита сам ключ). Но это лишь осложняло атаку грубой силой (brute force), остальных проблем это не решало, ключевой поток так же был подвержен коллизиям т.к. длина IV не изменилась. В 2004-ом году IEEE предложила улучшенную версию WEP2 со 128-битным IV, за счёт чего вероятность коллизий была сведена к минимуму. Но это не избавило протокол от основных недостатков, он остался подвержен статистическим атакам, используя которые можно узнать ключ всего за пару часов.

2. Выводы

Очевидно, применение WEP – это бесполезная трата вычислительной мощности и части трафика. Вместо этого лучше использовать проверенные временем решения, как IPSec или SSL. Сам акроним WEP потерял своё значение, и многие в шутку читают его как «Won't Even Protect».

3. Дополнительные материалы по теме

[1] Интернет сайт курса «Защита информации»: http://www.re.mipt.ru/infsec/index.html

[2] Антон Карпов «WEP. Антология уязвимостей и атак»: http://www.ank-pki.ru/wep/wep_ank.html

[3] IEEE 802.11 design sheet: http://standards.ieee.org/getieee802/download/802.11-1999.pdf

[4] RC4 description: http://en.wikipedia.org/wiki/RC4

[5] CRC32 description: http://en.wikipedia.org/wiki/CRC32

[6] Galois Field: http://en.wikipedia.org/wiki/Galois_field

[7] Bill Arbaugh, University of Maryland. «An Inductive Chosen Plaintext Attack against WEP/WEP2»: http://grouper.ieee.org/groups/802/11/Documents/DocumentHolder/1-230.zip

[8] Andrea Bittau and Mark Handley, University College London, Joshua Lackey, Microsoft. «The Final Nail in WEP's Coffin» (Conference Oakland'06): http://tapir.cs.ucl.ac.uk/bittau-wep.pdf

[9] N. Borisov, I. Goldberg, and D.Wagner. «Intercepting Mobile Communications: The Insecurity of 802.11» (Mobicom, Rome, Italy, July 2001): http://www.isaac.cs.berkeley.edu/isaac/mobicom.pdf

[10] Scott Fluhrer, Cisco Systems Inc., Itsik Mantin and Adi Shamir, The Weizmann Institute. «Weaknesses in the Key Scheduling Algorithm of RC4»: http://www.wisdom.weizmann.ac.il/~itsik/RC4/Papers/Rc4_ksa.ps

[11] Chopchop (Experimental WEP attacks): http://www.netstumbler.org/showthread.php?t=12489