

WLAN Security

1. WEP Problems

(C) Herbert Haas 2006/4/1

Content

In this chapter a detailed overview about today's WLAN security problems and solutions are presented.

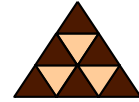
This subchapter provides an introduction into WEP, the basis of the 802.11 original and only method for encryption, authentication and integrity protection.

Objective

After completing this chapter the following tasks could be solved:

- Emphasize the basic vulnerabilities of WLAN
- Explain why WEP is insecure and give a mathematical example
- Compare WEP and TKIP/MIC
- Highlight the design flaws of the WLAN standard authentication
- Explain the design idea of 802.1x
- Compare EAP-TLS, LEAP, PEAP, EAP-TTLS, EAP-FAST with each other and emphasize important security features
- Explain the design concept of WPA and WPA2
- Implement a reliable 802.1x infrastructure over a WAN connection
- List important issues to be considered when choosing a VPN design
- Explain PSPF

Intro



- **Wireless LAN is a perfect media for attackers**
 - ♦ Sniffers easily remain undetected
 - ♦ Outdoor attacks
 - ♦ Simple DoS attacks through jamming
- **Vulnerabilities found in initial standards**
 - ♦ Authentication / Encryption / Integrity
 - ♦ Centralized management of user credentials
- **“Mobile devices” => frequent hardware theft**
- **Rogue APs often remain undetected**
 - ♦ Mutual auth required
- **Interoperability of security features of different vendors still in question (nevertheless WPA)**
- **Lots of cracker tools available (WEPCrack, AsLeap, ...)**
- **2002/2003: 66% of WLANs unprotected (but better security awareness in 2004)**

Compared to all other physical communication media, the wireless realm is the **best-of-choice medium for attackers** and hackers. The main reason for this is, that there are no wires an attacker needs to attach to. Moreover, the attacker can hide in another building or in a car, more than 100 meters outside the building (if he/she has a good antenna).

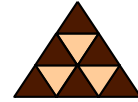
Since there are no wires it is not possible to protect the physical media from interferences or jamming, therefore **DoS attacks** are critical. An attacker could even destroy the sensitive receiving devices by jamming at very high power levels.

Additionally, there are other problems, caused by the 802.11 design itself:

- The standard encryption, integrity and authentication method has serious **design flaws**.
- There is no means to **manage user credentials** in a central way, which leads to bad practical security designs.
- The standard security concept is based on **device-bound secrets**, therefore hardware theft opens security holes for that network.
- The standard security concept does not allow to authenticate the infrastructure devices, therefore so-called "**rogue access points**" can be installed by attackers.
- Proprietary security enhancements caused an **interoperability problem** for several years.
- Dozens of **cracker tools** are available on the Web.

And finally, the WLAN **security awareness** only became widespread in the last year and still too many WLAN networks are poorly secured or not secured at all. In 2002/2003, almost two-third of all WLAN networks were unprotected.

RC4 Facts



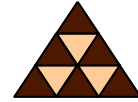
- **Simple and fast** stream cipher
 - ◆ Variable key lengths (1-256 bytes)
 - ◆ 15 times faster than 3DES
 - 8-16 operations per output byte
 - ◆ Also used by SSL/TLS
- Designed 1987 by **Ron Rivest** for RSA Security
 - ◆ Kept as trade secret by RSA Security but leaked out in 1994
- **Period is larger than 10^{100} !!!**

The security of stream ciphers depends 1) on the pseudo-randomness of the keystream they produce, and 2) of the implementation which must guarantee that each keystream is only used once! Since encryption and decryption is the same operation (XOR), if two plaintexts are encrypted with the same keystream, cryptanalysis is typically simple (for example, assume that one plaintext is known).

A stream cipher can be as secure as block cipher of comparable key length but typically stream ciphers are much faster and use far less code. For example, if 3DES can produce 3 Mbit/s on a Pentium II, then RC4 could achieve 45 Mbit/s, which is 15-times faster!

The RC4 algorithm had been kept as a trade secret by RSA Security, but in September 1994 the code was anonymously posted in the Cypherpunks mailing list.

How RC4 Works



```
for i = 0 to 255 do
  S[i] = i;
  T[i] = K[i mod keylen];
```

Initialize S[0]..S[255] with ascending numbers.
Initialize T[0]..T[255] with the key K (if keylen < 256 then repeat K as often as necessary).

```
j = 0;
for i = 0 to 255 do
  j = (j + S[i] + T[i]) mod 256;
  Swap (S[i], S[j]);
```

Use T to produce initial permutation of S.
Hereby go from S[0] to S[255] and swap each S[i] with another byte dictated by T[i].

After that, S still contains all numbers from 0 to 255 but in a permuted order.

```
i, j = 0;
while (1)
  i = (i + 1) mod 256;
  j = (j + S[i]) mod 256;
  Swap (S[i], S[j]);
  t = (S[i] + S[j]) mod 256;
  k = S[t];
```

Now again swap S[i] with another byte in S, but this time it is dictated by S itself (the key is no longer used).

After S[255] is reached, repeat again with S[0], as long as there are bytes to encrypt or decrypt.

XOR byte k with plaintext byte or ciphertext byte for encryption or decryption respectively.

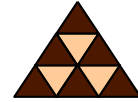
Possible key lengths range from 1 to 256 bytes (i. e. 8 to 2048 bits).

General Stream Cipher Issues



- Every stream cipher is supposed to produce a good pseudorandom "keystream"
 - ◆ This is the idea of a "one-time pad"
- The keystream is XORed with the plaintext
- This method is secure *if*
 - ◆ The keystream-generator has high entropy (i. e. really random)
 - ◆ **Each keystream is only used once**

Wired Equivalent Privacy (WEP)



- **Only encryption method of the 802.11 standard**
 - ♦ Used for privacy, integrity and authentication
- **Shared key method**
 - ♦ Either one static key
 - ♦ Or short list of dynamic keys (up to four)
- **Key lengths:**
 - ♦ 40 bit (default, aka "64 bit" with IV)
 - ♦ Optionally 104 (or "128" bit with IV)
- **No key distribution method defined(!)**

The Wired Equivalent Privacy (WEP) algorithm should provide a nearly-wired privacy look-and-feel, as its name suggests.

WEP uses the **RC4** PRNG algorithm from RSA Data Security Inc. RC4 is a stream cipher, a well studied algorithm, which expands a key into an infinite pseudorandom sequence.

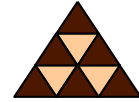
This RC4 key consists of a **40 bit or 104 bit secret key** and a **24 bit Initialization Vector (IV)**.

Note: The 40- or 104-bit WEP key is used as the base key for each packet. When combined with the 24-bit initialization vector, it is sometimes called the "**WEP seed**". Therefore WEP seeds are made of 64 or 128 bits in total and many manufacturers refer to the 104-bit WEP keys as 128-bit keys for this reason.

Unfortunately the IEEE 802.11 standard does not specify methods how to distribute the WEP keys to infrastructure and client devices.

Typically, most vendors allow to specify **up to four WEP keys** which can be dynamically chosen in order to confuse attackers.

Basic Principle



- **Payload is XORed with a RC4-generated pseudorandom **keystream K****
 - ♦ S depends on shared key and 24 bit Initialization Vector (IV)
 - ♦ Ciphertext $C = \text{Plaintext } P \oplus \text{Keystream } K$

Both the visible initialization vector and the shared secret WEP key are used by the RC4 algorithm to produce a pseudo-random **keystream** for encryption and decryption.

This keystream is mixed with the payload using the **XOR operation**. In principle the RC4 encryption is very secure—if there were no severe design flaws.

The weaknesses within WEP were first exposed by researchers from Intel, the University of California at Berkeley, and the University of Maryland. The most damning report came from Fluhrer, Mantin, and Shamir, which outlined a passive attack that Stubblefield, Ioanndis, and Rubin at AT&T Labs and Rice University implemented by capturing a hidden WEP key based on the attacks proposed in the Shamir et al. paper (aka **Fluhrer et. al.** paper). This attack took just hours to implement.

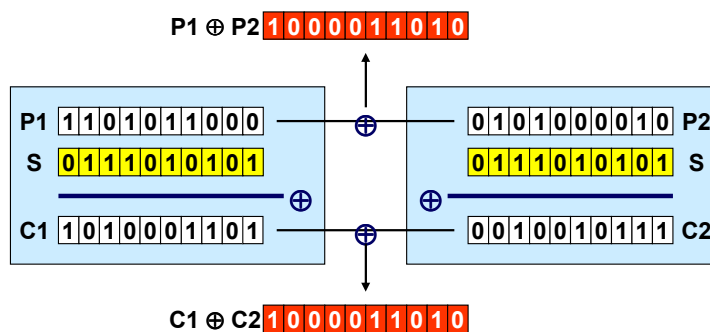
Ron Rivest, inventor of the RC4 algorithm, recommends that

"Users consider strengthening the key scheduling algorithm by preprocessing the base key and any counter or initialization vector by passing them through a hash function such as MD5. Alternatively, weaknesses in the key scheduling algorithm can be prevented by discarding the first 256 output bytes of the pseudo-random generator before beginning encryption. Either or both of these techniques suffice to defeat the [Fluhrer, Martin, and Shamir] attacks on WEP."

WEP – Design Flaw in Detail



- **The Problem:**
 - ♦ **XOR operation eliminates two identical terms!**
 - ♦ **If same S is used on different plaintexts, then**
 - $C1 = S \oplus P1$ and $C2 = S \oplus P2$
 - $C1 \oplus C2 = P1 \oplus P2$
 - Same keystream S cancels out!
 - ♦ **If P1 is known then P2 can be easily calculated!**



(C) Herbert Haas 2006/4/1

8

Although RC4 is a very good algorithm, its application with WEP reveals some remarkable security flaws. WEP is insecure when the **same keystream is used more than once**—the key length and the random properties of the keystream do not matter at all!

This is because the **XOR operation eliminates two identical terms**. That is, if an attacker sniffed Ciphertext C1 and Ciphertext C2, which had been produced by the same keystream S, then actually the following operations were made by the WEP algorithm:

$$C1 = S \oplus P1 \text{ and } C2 = S \oplus P2.$$

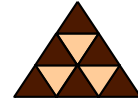
Hence $C1 \oplus C2$ cancels out S and equals $P1 \oplus P2$. Thus, if Plaintext P1 is known, P2 can be easily calculated!

Note: This attack method also works for a subset of these "vectors": If a part of P1 is known, then a congruent part of P2 can be calculated.

Knowledge of parts of the plaintext message can enable **statistical attacks** to recover all plaintexts. These statistical attacks become increasingly practical as more ciphertexts that use the same key stream are known. Once one of the plaintexts becomes known, it is trivial to recover all of the others.

Although most 802.11 equipment is designed to disregard encrypted content for which it does not have the key, it is relatively simple to change the configuration of the drivers. Active attacks, which requires transmission seems to be more difficult, yet not impossible. Many 802.11 products come with programmable firmware, which had been reverse-engineered and modified to provide the ability to inject traffic to attackers.

IV Collisions



- **Keystream should change for each packet**
 - ♦ Assures that same plaintexts result in different Ciphertext
 - ♦ 802.11 does not specify how to pick IVs
 - ♦ Many implementations reset IV to zero at startup and then count up
- **Only 2^{24} IV choices → Collisions will occur !!!**
 - ♦ Attacker could maintain a "codebook" of all possible S
 - ♦ $1500 \text{ byte} \times 2^{24} = 24 \text{ GByte}$
 - ♦ Matter of hours only
- **Shared key length does not hamper the attack!**

Because of the XOR properties it is crucial to continuously change the key that makes up the particular keystream—ideally for each packet sent! The key is made up of the shared secret and the IV, and the latter was intended to assure collision protection. But actually, **the standard does not specify how to change the IV**. There is no strict requirement to change IVs at all!

Example of an attack duration:

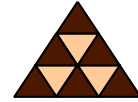
A busy access point, which constantly sends 1500 byte packets at 11Mbps, will exhaust the space of IVs after $1500 * 8 / (11 * 10^6) * 2^{24} = \sim 18000$ seconds, or 5 hours. This allows an attacker to collect two Ciphertexts that are encrypted with the same key stream and perform statistical attacks to recover the plaintext.

Now it is clear, that the shared **key length** do not affect this sort of attack at all (also see Jesse Walker's "Unsafe at any key length" paper). If P1 is known then P2 is immediately available. Much of network traffic contains predictable information, but it is much easier when three or more packets collide. Certain devices on the market utilize the IV in a simply **predictable** way, for instance by incrementing by one for each packet. Furthermore, the IV value is reset at each startup.

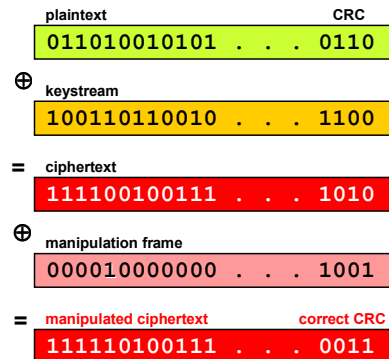
One New York computer security consultant who was quoted in the Wall Street Journal article says he was able to access the computer network of his client, a major financial services firm on Wall Street, while sitting on a bench across the street.

Common wireless sniffing tools are **WEPcrack** and **AirSnort**.

Integrity Vulnerability



- Encrypted CRC is used to check integrity
- But **CRC is linear**:
 - ◆ $CRC(X \oplus Y) = CRC(X) \oplus CRC(Y)$
- Thus payload bits can be manipulated, because
 - ◆ $RC4^K(X \oplus Y) = RC4^K(X) \oplus Y$
 - ◆ $RC4^K(CRC(X \oplus Y)) = RC4^K(CRC(X)) \oplus CRC(Y)$
- Attacker can easily modify known bytes of packets (at least L3/L4 header structures are known)



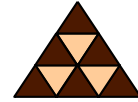
Furthermore, WEP is also used to protect the integrity of a frame in combination with the CRC. But the CRC is a **linear operation** and can therefore be **additively decomposed**.

Because of this property, an attacker could XOR a plaintext X with another plaintext Y for manipulation purposes and only has to calculate $CRC(X) \oplus CRC(Y)$ to get $CRC(X \oplus Y)$. Because of the linearity, this operation can also be successfully applied even when the CRC is RC4-encrypted!

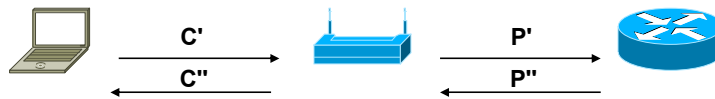
Thus the “Integrity check” does not prevent packet modification, and an attacker can **easily flip bits in packets**, modify active streams, or bypass access control.

Even partial knowledge of the packet is sufficient if the attacker wants only to modify the known portion.

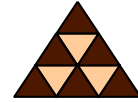
Bit-Flipping Attack Example



- Attacker catches and manipulates encrypted frame, updates ICV
- AP decrypts frame, validates ICV and forwards frame
- Router detects fault and sends predictable error message
- $\text{Keystream} = C'' + P''$



Attacks Overview



- **Keystream reuse (IV collisions)**
 - ♦ Dictionary-building attacks
 - ♦ Allows real-time automated decryption of all traffic
- **Bit-flipping attacks**
 - ♦ Attacker intercepts WEP-encrypted packet, flips bits recalculates CRC and retransmits forged packet to AP with same IV
 - ♦ Because CRC32 is correct, AP accepts and forwards frame
 - ♦ Layer 3 end device rejects and sends a predictable response
 - ♦ AP encrypts response and sends it to attacker
 - ♦ Attacker uses response to derive key
- **Fluhrer, Mantin, Shamir (FMS) attack on RC4**
 - ♦ RC4 key scheduling is insufficient
 - The beginning of the pseudorandom stream should be skipped, otherwise some IV values reveal information about the key state
 - ♦ Key can be recovered after several million packets
- **KoreK Attack**
 - ♦ Packet manipulation, reinjection and CRC analysis
 - ♦ Key can be recovered after several 100,000 packets
- **Arbaugh Attack**
 - ♦ Calculate arbitrary additional bytes on a known but short keystream

The presented WEP attacks only belong to the most simple one. Here is a summary of the most practical attack methods.

Keystream reuse attack:

This already described method is typically combined with dictionary-building and statistical analysis. Finally the attacker has created a large dictionary containing all keystreams possible with the used WEP keys and then he/she can perform real-time decryption of all traffic.

Bit flipping attacks:

The attacker could make guesses about the headers of a packet, which contains typically a lot of redundancy that is predictable. In particular, all that is necessary to guess is the destination IP address. Now the attacker can flip appropriate bits to transform the destination IP address to send the packet to another machine, which is in his own realm. Most wireless networks are connected to the Internet and the APs will decrypt each packet that is destined to a wired destination. This is also called a redirection attack.

If a guess can be made about the TCP headers of the packet, the attacker could change the destination port to be port 80, which will allow it to be forwarded through most firewalls. Note that the IP checksum can be easily spoofed and the TCP checksum is disregarded by the network.

Changing an IP address is relatively simple. Assume the high and low 16-bit words of the original IP address are IP_H1 and IP_L1, and should be changed to IP_H2 and IP_L2. If CRC1 is the original IP checksum, then $CRC2 = CRC1 + IP_H2 + IP_L2 - IP_H1 - IP_L1$ in one's complement math.

If the attacker knows CRC1 by some means, he then figures out CRC2 as above and computes CRC1 XOR CRC2 to get to the final checksum. Another way is to make guesses about the IP address and see if they work. The TCP reaction attack works by seeing what the reaction of the system is to forgeries. A correctly guessed IP will be accepted by the system, while a bad one causes the packet to be dropped into the bit bucket. This only works on TCP packets, because the attacker needs the ACKs that TCP sends (the TCP ACK packet is of a standard size) when the TCP checksum is correct.

Fluhrer et. al. attack:

Some IV values reveal information about key state, thus the shared keys can be recovered after several million packets. In the RC4 algorithm the Key Scheduling Algorithm (KSA) creates an IV based on the base key. A flaw in the WEP implementation of RC4 allows "weak" IVs to be generated. The RC4 key scheduling is insufficient: the beginning of the pseudorandom stream should be skipped.