

WLAN Security Introduction

Threats Overview

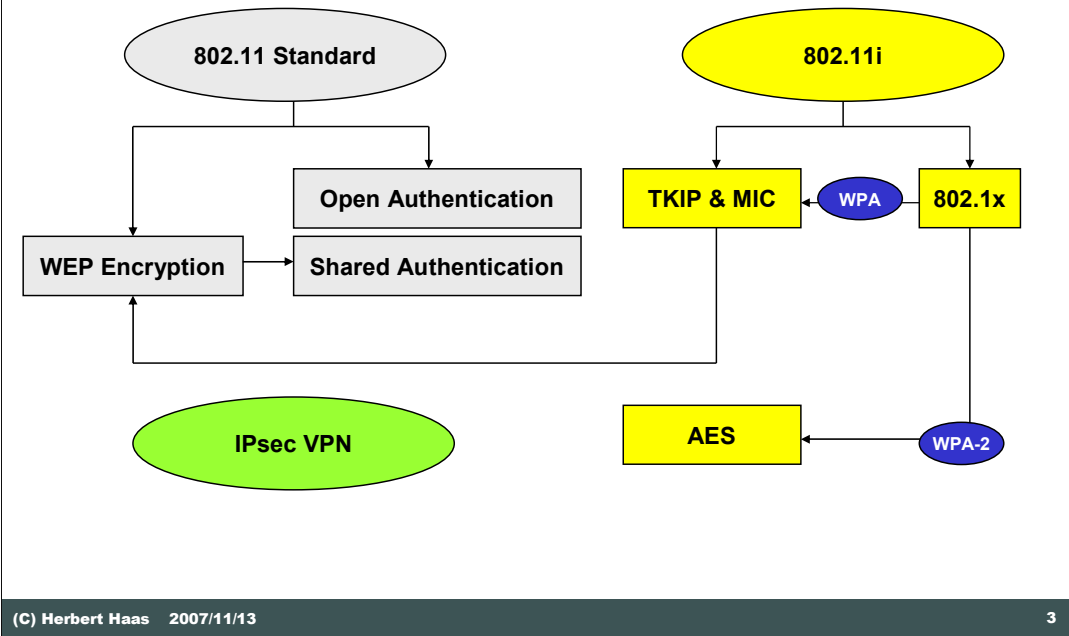
(C) Herbert Haas 2007/11/13

Threat Summary



- **Simple eavesdropping**
 - ◆ Radio broadcast
 - ◆ Reduce TX powers!
 - ◆ Encryption (WEP, TKIP, AES, IPsec)
- **Authentication**
 - ◆ Shared secrets vs. stolen devices, large nets
 - ◆ Centralized AAA => 802.1x
 - ◆ Mutual authentication (Rogue APs)
- **DoS Attacks**
 - ◆ Physical jamming
 - ◆ Difficult to prevent (shielding, directional antennas)

WLAN Security Overview



WEP Problems

(C) Herbert Haas 2007/11/13

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.

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.

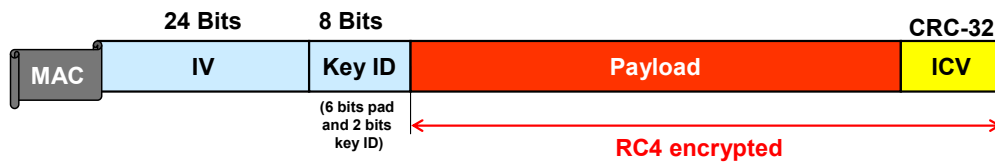
This RC4 key consists of a **40 bit or 104 bit secret key** and a **24 bit Initialization Vector (IV)**.

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.

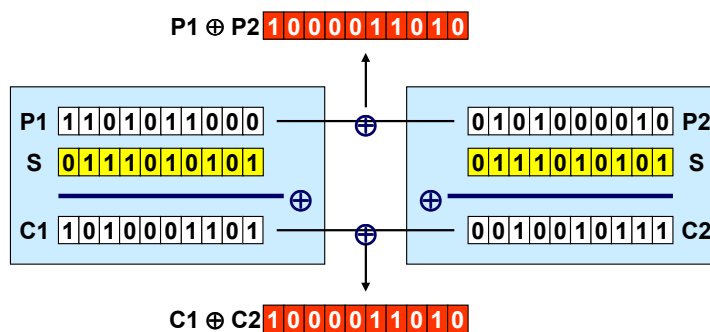
Ron Rivest, inventor of the RC4 algorithm, recommends that

"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 2007/11/13

11

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)

plaintext	CRC
011010010101 . . .	0110
⊕	
keystream	
100110110010 . . .	1100
=	
ciphertext	
111100100111 . . .	1010
⊕	
manipulation frame	
000010000000 . . .	1001
=	
manipulated ciphertext	correct CRC
111110100111 . . .	0011

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!

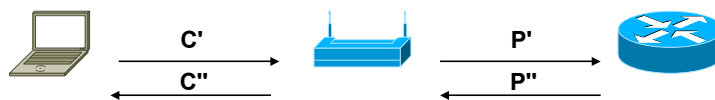
Thus the “Integrity check” does not prevent packet modification, and an attacker can **easily flip bits in packets**, modify active streams, or bypass access control.

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
- $\text{Keystream} = C'' + P''$



Arbaugh Attack



- **Allows to arbitrarily expand a known keystream of size n**
 - ◆ Easily done with known messages (e. g. DHCP discoveries)
- **Create messages of size $n-3$ and encrypt it with the known keystream**
- **Only the last byte (4th CRC byte) is not encrypted: trial and error!**
- **On average only 128 trials necessary for every additional byte!**

The Arbaugh Attack

Here is a more detailed example to understand the Arbaugh attack:

1. Find an initial keystream S of size n . For example look for DHCP-Discover messages, which have a fixed size and a broadcast MAC destination address. The known plaintext of the DHCP-Discover message consists of a source IP address of 0.0.0.0, a broadcast destination IP address 255.255.255.255, and some other fixed header information. This method reveals 24 bytes of keystream, that is $n = 24$.
2. Create a message M of size $n - 3$, that is 21 bytes in our case. For example an ARP request or an ICMP packet.
3. Create the ICV of the message M and append three bytes of it to the message, resulting in a plaintext P .
4. XOR the known keystream to the plaintext: $C = P \text{ XOR } S$.
5. Instead of the true fourth byte of the ICV append a test byte B_i to the ciphertext C . For example the first test byte could be $B_0 = 0x00$. The resulting ciphertext packet is C_i .
6. Send C_i to the AP. If the last byte B_i (i. e. the fourth byte of the ICV) was correctly encrypted then the AP accepts the packet and the network will send a response. If B_i was wrongly encrypted then the AP will discard the packet silently. Next try $B_1 = 0x01$, $B_2 = 0x02$, ... , $B_{255} = 0xFF$. On average after 128 trials B_i is found.
7. Since the whole ICV is known as plaintext, calculate the unknown keystream-byte $S_{25} = B_i \text{ XOR } ICV_4$. Remember that $B_i = ICV_4 \text{ XOR } S_{25}$.

Practically one could create an ICMP echo request of increasing length. If the frame has been correctly encrypted then there will be an ICMP echo reply. (Remember that the payload of an ICMP packet may have arbitrary length.)

Attacks Summary (1)



- **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

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.

Attacks Summary (2)



- **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
 - ◆ 'WEPplus' = WEP with avoidance of weak IVs
- **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

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.

The **KoreK Attack** was first implemented in the tool "ChopChop" and is now part in nearly all WEP cracking tools, such as aircrack or aircrack-ng.

Also the **Arbaugh Attack** is an acceleration tool and therefore part in many modern WEP cracking tools.

Interim Solutions: TKIP and MIC

(C) Herbert Haas 2007/11/13

Content

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

This subchapter provides an introduction into TKIP and MIC.

802.11i



- **Two new network types**
 - ♦ **Transition Security Network (TSN)**
 - ♦ **Robust Security Network (RSN)**
- **An RSN only allows devices using TKIP/Michael and CCMP**
- **A TSN supports both RSN and pre-RSN (WEP) devices**
 - ♦ **Problem: broadcast packets have to be transmitted with the weakest common denominator security method**
 - ♦ **Consider a single client only supporting WEP**

Task Group i (TGi) was formed in March 2001 as a split from the MAC Enhancements Task Group (TGe). Its charge was to "enhance the 802.11 Media Access Control (MAC) to enhance security and authentication mechanisms." TGi finished work on the 802.11i standard, and it has been approved.

802.11i defines two WLAN network types: Transition Security Network (TSN) and Robust Security Network (RSN). RSNs only allow devices which support TKIP/Michael and CCMP. TSNs support both RSN devices and legacy pre-RSN, i. e. WEP devices. The drawback with RSN is that broadcast packets have to be transmitted with the weakest common denominator security method. If there is a device using WEP in a TSN network, it weakens the security of broadcast traffic for all the devices. RSN is definitely preferred, and getting all networks to use CCMP exclusively is the long term goal.

802.11i



**Pre-standard
802.11i
(WPA)**

- **Message Integrity Check (MIC)**
 - ◆ Nonlinear algorithm
- **Temporal Key Integrity Protocol (TKIP or “WEP2”)**
 - ◆ Also uses RC4-based WEP without the known flaws
 - Per-packet keys through IV mixing
 - Replay protection
 - ◆ Essentially a patch for WEP

**Ratified 802.11i
(WPA2)**

First WPA2 certifications
already since 1st Sept 2004

- **Counter Mode CBC MAC (CCMP)**
 - ◆ = AES + CBC-MAC
 - ◆ Replaces WEP !!!
(requires new HW support)

Recently, the **IEEE 802.11i** Security Task Group released two "informative texts" providing WEP hardening: MIC and TKIP. The IEEE 802.11 Task Group "i" is working on standardizing WLAN encryption improvements. Two new network types, called Transition Security Network (TSN) and Robust Security Network (RSN) had been defined.

The Temporal Key Integrity Protocol (TKIP), initially referred to as WEP2) is an interim solution (as part of TSN) that fixes the key reuse problem of WEP.

TKIP is a compromise on strong security and possibility to use existing hardware. Still uses RC4 but per-packet keys plus replay protection through a keyed packet authentication mechanism (Michael MIC).

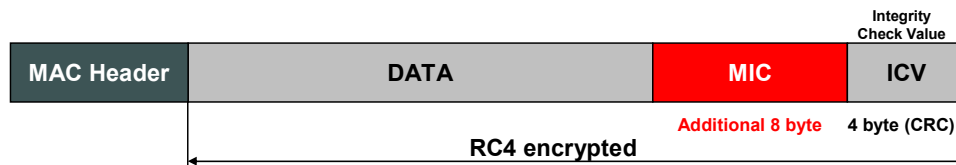
TKIP begins with a 128 bit "temporal key" shared among clients and access points. TKIP combines the temporal key with the client's MAC address and then adds a 6-byte IV to produce the key that will encrypt the data. Thus each station uses different key streams for encryption. TKIP changes keys every 10,000 packets, using a dynamic distribution method.

The IEEE plans to use the **Advanced Encryption Standard (AES)** instead of RC4 for TKIP in the long run (RSN), combined with Counter Mode - Cipher Block Chaining - Message Authentication Code (CBC MAC) to provide strong integrity and message authentication. Also the term "Wireless Robust Authenticated Protocol" (WRAP) is sometimes used synonymously for this concept.

The **Wi-Fi** specified TKIP and MIC as mandatory features of the **Wi-Fi Protected Access (WPA)** protocol, while AES should be part of WPA2.

Note: WiFi and particular vendors uses **different** TKIP/MIC algorithms, which are not compatible. Even WPA was intended to be an intermediate solution because the WiFi only picked a subset of the IEEE 802.11i working draft 3.0).

MIC (as used by WPA)



- **Encrypted checksum**
 - ♦ => Nonlinear function now
- **Uses "Michael" algorithm**
 - ♦ Much more lightweight than MD5 or SHA
- **Uses separate 64-bit key**
 - ♦ Data Integrity Key (DIK) derived from PTK after WPA key management
 - ♦ AP and STA use different MIC keys (128-bit DIK is split)

The Message Integrity Check (MIC) provides data integrity similar to CRC but provides a **non-linear** operation, the "Michael" algorithm, and is therefore not vulnerable after RC4 encryption.

The MIC is based on a **seed** value or a **secret key**, the destination and source MAC, and payload. That is, any change of these values significantly alter the MIC.

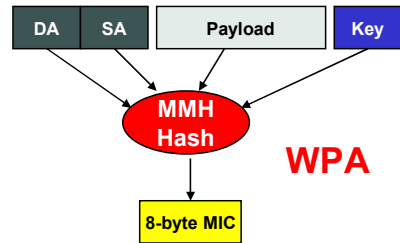
The 802.11i task group felt that other commonly used hashing algorithms such as SHA-1 were too computation-intensive to calculate on legacy hardware, so they agreed on the simpler Michael algorithm. Like many hash algorithms, Michael is calculated over the length of the packet, but all of the scrambling it does is based on shift operations and XOR additions, which are quick to calculate. Michael uses a key called the Michael key, which is derived during the WPA procedure (pairwise key).

But according to the 802.11i specification, the Michael algorithm "provides only weak protection against active attack." Therefore **MIC countermeasures** have been specified by the 802.11i: 1) logging and 2) disable and deauthenticate. If two Michael failures occur within one minute, both ends should disable all packet reception and transmission. In addition, the AP should deauthenticate all stations and delete all security associations—a rather drastic solution.

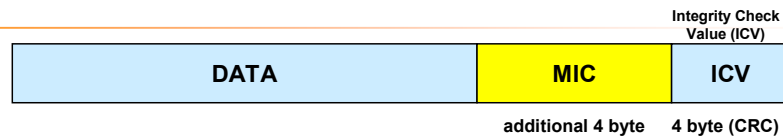
MIC Problems



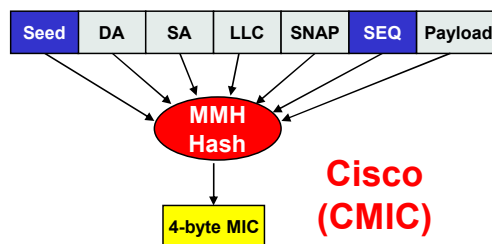
- **Michael algorithm**
 - ◆ Provides security level of only 20 bit strength
 - ◆ Attacker can construct forgery after approx 2^{19} tries (520,000 frames)
- **MIC Countermeasures**
 - ◆ Upon two MIC failures within 60 seconds, this AP disassociates *all* stations for at least 60 seconds and erases current keys in use
 - ◆ So attacker forgery trials become nearly impossible
 - ◆ Typically turned OFF (DoS!!!)



Cisco MIC (CMIC)

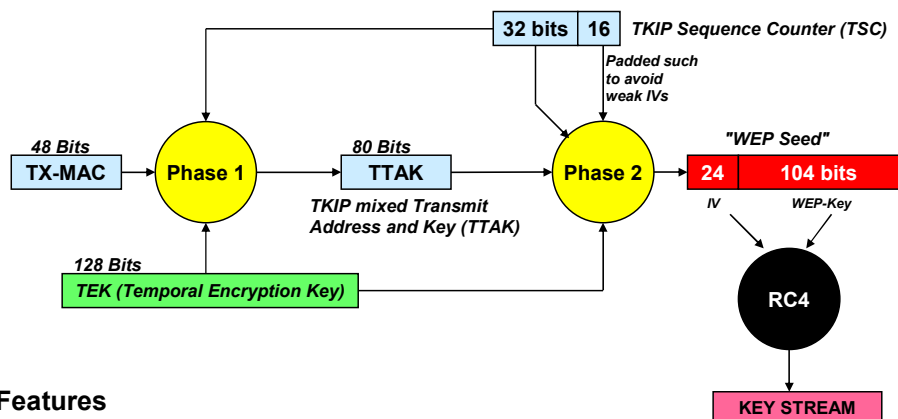


- Uses a seed value as pseudo-key
- Uses sequence number (AP verifies order)



Note: The Cisco Message Integrity Check serves the same purpose as the 802.11i MIC and is in fact stronger than Michael. It is based on Shai Halevi and Hugo Krawczyk's MMH hashing algorithm.

TKIP (As used by WPA)



- **Features**
 - ♦ Longer and unpredictable IV through IV/key mixing
 - ♦ Encrypted replay protection number (TSC)
- **WPA TKIP**
 - ♦ 48 bit IV, includes MAC
 - ♦ Fast S-box mixer
 - ♦ Fresh session keys on every association

The WPA's TKIP solution complies to the 802.11i proposals and uses fresh session keys on every association as well an 48-bit IV space. The mixing functions are based on substitution boxes (S-boxes), which are computationally very efficient, compared to other hash functions.

The **Temporal Encryption Key (TEK)** is derived from the "**Pairwise Master Key**" (PMK, also called "base key"), which has been negotiated by the WPA key management protocol. The TEK is used to securely hash a packet counter, the **TKIP Sequence Counter (TSC)**, and the transmit MAC address. A second hash stage enhances the security of the S-box principle.

The TSC is split into 16-bit and 32-bit parts. The 16-bit part is padded to 24 bits to produce a traditional IV. The padding is done in a way that avoids the possibility of weak IV generation. Interestingly, the 32-bit part is not used for the transmitted IV generation; instead, it is utilized in the TKIP per-packet key mixing.

Phase 1 eliminates the use of the same key by all connections, and the second phase reduces the correlation between the IV and per-packet key.

The TSC starts at 0 and increases by 1 for each packet. TSCs must be remembered because they must never repeat for a given key. Each receiver keeps track of the highest value it has received from each MAC address. If it receives a packet that has a TSC value lower than or equal to one it has already received, it assumes it is a rebroadcast and drops it. Thus, packets can only arrive in sequence.

TKIP is only a SW-addon and can reuse the existing WEP hardware.

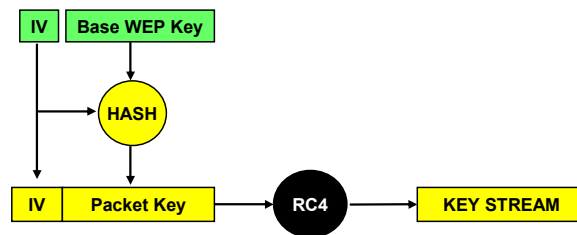
TKIP Details



- **Phase 1**
 - ♦ The high-order 32 bits of the TSC are combined with the TA and the first 80 bits of the TEK.
 - ♦ This phase of the key mixing is an iteration involving inexpensive addition, XOR, and AND operations, plus an S-box lookup reminiscent of the RC4 algorithm. These were chosen for their ease of computation on low-end devices such as APs.
 - ♦ Phase 1 produces an 80-bit value called TKIP mixed Transmit Address and Key (TTAK). Note that the only input of this phase that changes between packets is the TSC. Because it uses the high-order bits, it only changes every 64K packets.
 - ♦ Phase 1 can thus be run infrequently and use a stored TTAK to speed up processing. The inclusion of the transmitter's MAC address is important to allow a pair of stations to use the same TEK and TSC values and not repeat RC4 keys.
- **Phase 2**
 - ♦ Now the TTAK from phase 1 is combined with the full TEK and the full TSC.
 - ♦ This phase again uses inexpensive operations, including addition, XOR, AND, OR, bit-shifting, and an S-box.
 - ♦ The output is a 128-bit WEP seed that will be used as the RC4 key in the same manner as traditional WEP.
 - ♦ In the phase 2 algorithm, the first 24 bits of the WEP seed are constructed from the TSC in a way that avoids certain classes of weak RC4 keys.

BTW: TKIP was designed as a 5 year interim solution only! Obviously it will be used much longer than intended.

Cisco TKIP ("CKIP")



- **Simple proprietary solution**
- **Still uses 24 bit IV but calculates per-packet WEP keys from IV**
 - ◆ **Hash-based mixer**

Because urgent security demands of the market, Cisco developed a proprietary "**Cisco KIP**" (**CKIP**), which is based on hashing the static WEP key together with the 24-bit IV to gain the actual packet key.

Also Cisco's solution provides per-packet keys, but **it is recommended to use WPA's TKIP** because:

- WPA's TKIP is computationally more efficient.
- It is more secure, because of the PMK involved.
- The dynamical RC4-key space is much bigger as compared to CKIP.
- Nearly all important vendors support WPA.



- **Against rumors, TKIP is reasonably safe!**
 - ♦ For each packet, the 48-bit IV is mixed with the 128-bit PTK to create a 104-bit RC4 key
 - There is practically no statistical correlation
 - Estimated one weak-IV per century (!)
 - ♦ Countermeasures against traffic re-injection
 - Sequence numbers + MIC
 - ♦ Robust 4-way handshake
- **Only problem: WPA-PSK**
 - ♦ Which uses a specified passphrase to PMK mapping => good passphrase required !!!
 - ♦ Otherwise dictionary attack possible

The estimated weak IV frames appearance interval with TKIP is about a century, so by the time a cracker collects the necessary 3,000 or more interesting IV frames, he or she would be 300,000 years old. [Found somewhere: CHECK!]

AES and CCMP

(C) Herbert Haas 2007/11/13

Content

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

This subchapter provides an introduction into AES and CCMP.

802.11i



Pre-standard
802.11i – TSN
(WPA)

- **Message Integrity Check (MIC)**
 - ◆ Nonlinear algorithm
- **Temporal Key Integrity Protocol (TKIP or “WEP2”)**
 - ◆ Also uses RC4-based WEP without the known flaws
 - Per-packet keys through IV mixing
 - Replay protection
 - ◆ Essentially a patch for WEP

Ratified 802.11i
– RSN
(WPA2)

First WPA2 certifications
already since 1st Sept 2004

- **Counter Mode CBC MAC (CCMP)**
 - ◆ = AES + CBC-MAC
 - ◆ Replaces WEP !!!
(requires new HW support)

Recently, the **IEEE 802.11i** Security Task Group released two "informative texts" providing WEP hardening: MIC and TKIP. The IEEE 802.11 Task Group "i" is working on standardizing WLAN encryption improvements. Two new network types, called Transition Security Network (TSN) and Robust Security Network (RSN) had been defined.

The Temporal Key Integrity Protocol (TKIP), initially referred to as WEP2) is an interim solution (as part of TSN) that fixes the key reuse problem of WEP.

TKIP is a compromise on strong security and possibility to use existing hardware. Still uses RC4 but per-packet keys plus replay protection through a keyed packet authentication mechanism (Michael MIC).

TKIP begins with a 128 bit "temporal key" shared among clients and access points. TKIP combines the temporal key with the client's MAC address and then adds a 6-byte IV to produce the key that will encrypt the data. Thus each station uses different key streams for encryption. TKIP changes keys every 10,000 packets, using a dynamic distribution method.

The IEEE specifies to use the **Advanced Encryption Standard (AES)** instead of RC4 for TKIP in the long run (RSN), combined with Counter Mode - Cipher Block Chaining - Message Authentication Code (CBC MAC) to provide strong integrity and message authentication. Also the term "Wireless Robust Authenticated Protocol" (WRAP) is sometimes used synonymously for this concept.

The **Wi-Fi** specified TKIP and MIC as mandatory features of the **Wi-Fi Protected Access (WPA)** protocol, while AES should be part of WPA2.

Note: WiFi and particular vendors uses **different** TKIP/MIC algorithms, which are not compatible. Even WPA was intended to be an intermediate solution because the WiFi only picked a subset of the IEEE 802.11i working draft 3.0).

WPA2 aka 802.11i



- **Exactly the same as WPA1 except...**
 - ◆ CCMP (AES in counter mode) instead of RC4
 - ◆ HMAC-SHA1 instead of HMAC-MD5 for the EAPoL MIC
- **Against rumors WPA2 is only a LITTLE better than WPA1**
 - ◆ **But neither will be cracked in the near future !!!**

How secure is AES compared to RC4?

RC4 uses up to 128 bits key length, AES uses 256 bits, that is the AES key is 128 bits longer. If only brute force attacks are assumed (algorithms are save enough) and considering Moore's law (computing power doubles every 18 month), then AES is at least $\log_2(128) * 18$ months ahead, that is more than 10 years, compared to RC4.

802.11i: CCMP – Overview



- **AES for data encryption (privacy)**
 - ♦ 128-bit block cipher
 - ♦ No per-packet keying needed
 - ♦ HW-realization recommended
 - ♦ Key-life determined by 48-bit IV
- **AES requires a **feedback mode****
 - ♦ To avoid the risks associated with the trivial Electronic Codebook (ECB) mode
 - Repeating patterns are not hidden
 - Not recommended for messages longer than one block !
- **The IEEE is still deciding which feedback mode to standardize for AES encryption – two choices:**
 - ♦ **Counter Mode CBC MAC (CCM)**
 - Provides encryption, authenticity and integrity
 - Applied on both header and data
 - IV also used to prevent replay attacks
 - WLAN's current favourite
 - ♦ **Offline Code Book (OCB) mode**
 - Problem: patented
 - Also supported by some WLAN vendors

The **802.11i** standard was finished in May 2004 and **approved in June 2004**. The main result, WPA2, includes support for more robust encryption algorithm (CCMP: AES in Counter mode with CBC-MAC) to replace TKIP and **optimizations for handoff** (reduced number of messages in initial key handshake, pre-authentication, and PMKSA caching).

The **Advanced Encryption Standard (AES)** is considered as state-of-the-art encryption method, designed recently, using Rijndael as algorithm and is official successor of DES or 3DES. This 128-bit block cipher is considered unbreakable for the next ten years or so.

CCM is actually the block cipher *mode* of AES that provides both encryption and authentication. It is a combination of counter-mode encryption and CBC-MAC authentication which are two modes that have been studied extensively for many years. CCM was developed as a non-patented alternative to OCB ("Offset Codebook") for use in secure wireless networks, but it can be used in almost any situation that requires secure communications. With CCM encryption and authentication

Links:

Rijndael description and algorithm:

<http://csrc.nist.gov/CryptoToolkit/aes/rijndael/>

AES Lounge:

<http://www.iaik.tu-graz.ac.at/research/krypto/AES/>

Cipher Block Chaining (CBC)



- **No patent**
- **Encryption and MAC use different nonces**
 - ♦ Collision attacks possible but sufficient mitigation when key management provides frequent key changes
- **Identical ciphertext blocks result only when:**
 - ♦ Same key and
 - ♦ Same plaintext and
 - ♦ Same IV is used
- **CBC is self-synchronizing**
 - ♦ If an error (including loss of one or more entire blocks) occurs in block c_j but not c_{j+1} , then c_{j+2} is correctly decrypted to x_{j+2} .

1. Encryption: $c_0 \leftarrow IV$. For $1 \leq j \leq t$, $c_j \leftarrow E_K(c_{j-1} \oplus x_j)$.
2. Decryption: $c_0 \leftarrow IV$. For $1 \leq j \leq t$, $x_j \leftarrow c_{j-1} \oplus E_K^{-1}(c_j)$.

Although CBC mode decryption recovers from errors in ciphertext blocks, modifications to a plaintext block x_j during encryption alter all subsequent ciphertext blocks. This impacts the usability of chaining modes for applications requiring random read/write access to encrypted data.

An exposed IV might allow a man-in-the-middle (MITM) to change the IV value in-transit. Changing the IV changes only the deciphered plaintext for the first block, without garbling the second block. Any or all bits of the first block plaintext can be changed systematically with complete control.

The most obvious way to prevent deliberate MITM changes to the first block plaintext with the IV is to encipher the IV; that prevents an opponent from changing plaintext bits systematically.

Counter Mode (CCM)



- **Instead of directly encrypting the data only a counter is encrypted**
- **Message is then XORed with this encrypted counter**
- **Counter = nonce (SQNR, Source-MAC, Priority fields)**

WPA2 supports **FIPS 140-2** compliant security, basically AES in counter mode. (An early draft included AES-OCB instead but it was dropped due to patent issues.) A 48 bit IV protects against replay attacks.

Authentication and Integrity is maintained using an **8 byte CBC-MAC** with a 48 bit nonce. Besides the data also the source and destination MAC addresses in the header are protected by the CBC-MAC. (These fields are called Additional Authentication Data (AAD).

The CBC-MAC, the nonce, and additional 2 byte IEEE 802.11 overhead make the CCMP packet 16 octets larger than an unencrypted IEEE 802.11 packet.

The AP advertises cipher suites both in beacons and probe responses.

Offset Code Book (OCB)



- **Patented**
- **Combines authentication and encryption**
 - ♦ Slightly faster than CBC encryption
 - ♦ More prone to collision attacks than CBC-MAC
- **If a particular collision on 128-bit values occurs, then an attacker can modify the message without being detected by the OCB authentication function**
 - ♦ Weak authentication algorithm – uses same nonce for encryption and authentication
 - ♦ In order to limit the probability of a successful forgery attempt to less than 2^{-64} change the key after 2^{32} blocks of data
 - ♦ Indeed strong enough for many people but does not justify 128-bit AES as successor of DES

AES-OCB is a mode that operates by augmenting the normal encryption process by incorporating an offset value.

The routine is initiated with a unique nonce (the nonce is a 128-bit number) used to generate an initial offset value. The nonce has the XOR function performed with a 128-bit string (referred to as value L).

The output of the XOR is AES-encrypted with the AES key, and the result is the offset value.

The plain-text data has the XOR function performed with the offset and is then AES-encrypted with the same AES key.

The output then has the XOR function performed with the offset once again. The result is the cipher-text block to be transmitted.

The offset value changes after processing each block by having the XOR function performed on the offset with a new value of L.

See <http://www.cs.ucdavis.edu/~rogaway/ocb/index.html>

OCB Algorithm



Convention: Message M, Key K, Nonce N

Define $L := E_K(0)$ from which the offset $Z_i := \gamma_i \cdot L \oplus R$ follows.
 $R := E_K(N \oplus L)$

Then the message is split into M_1, \dots, M_m , where only M_m is typically a non-128 bit block. The messages M_1, \dots, M_{m-1} are encrypted as follows:

$$\begin{aligned} X_i &:= M_i \oplus Z_i \\ Y_i &:= E_K(X_i) \\ C_i &:= Y_i \oplus Z_i \end{aligned}$$

While M_m is encrypted using μ denoting the length of this block:

$$\begin{aligned} X_m &:= \mu \oplus x^{-1} \cdot L \oplus Z_m \\ Y_m &:= E_K(X_m) \\ C_m &:= M_m \oplus \text{first-}\mu\text{-bits}(Y_m) \end{aligned}$$

The authentication is performed in two steps:

$$\begin{aligned} S &:= M_1 \oplus \dots \oplus M_{m-1} \oplus C_m 0^* \oplus Y_m \\ T &:= \text{first-}\tau\text{-bits}(E_K(S) \oplus Z_m) \end{aligned}$$

$C_m 0^*$... last ciphertext block padded with zeros to full 128 bit length

... "Checksum"

... "MAC Tag" of arbitrary length, depending on security vs. transmission cost trade-off. Typically 32..80 (documentation)