WEP (Wired Equivalent Privacy)

Ana Hernández Rabal

16-11-2007

Índice

- Introducción
- Autenticación
- Cifrado
  - CRC32
  - RC4
- Vulnerabilidades WEP
- Ataques
- Conclusiones
- Líneas futuras

Introducción

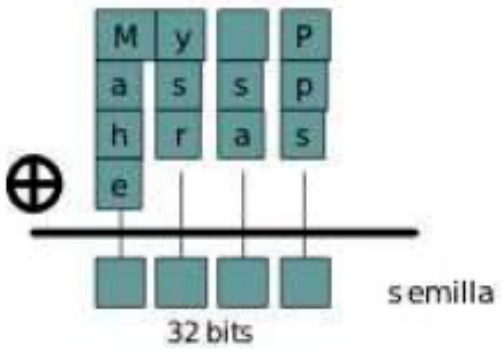
- WEP: Algoritmo opcional de seguridad incluido en el IEEE 802.11.
- Comprime y cifra los datos que se envían a través de las ondas de radio.
- Implementado por la capa MAC y soportado por la mayoría de los fabricantes. No compatible con IPSec.
- Objetivo: Dotar WLANs de seguridad similar a LANs
  - Principal: confidencialidad de los datos mediante cifrado.
  - Secundario: proporcionar autenticación. No explicito en el 802.11.

Autenticación

- Dos métodos:
  - OSA: Open System Authentication
    - Autenticación a cualquier estación que lo solicite.
  - SKA: Shared Key Authentication (Habilitado cifrado WEP)
    - STA envía Authentication request al AP.
    - AP envía desafío a la STA.
    - STA cifra el desafío y lo envía al AP.
    - AP descifra desafío y lo compara. Si es igual repuesta positiva, sino negativa.
- Con SKA se podría conseguir parte del flujo pseudoaleatorio (keystream) para un IV capturando el 2º y 3er paquete del proceso de autenticación. Mejor OSA.

Cifrado (1): Generación de llave

- Clave o passphase normalmente estática conocida por los clientes.
- XOR con la passphrase (ASCII) obteniendo semilla de 32 bits



M	y		P	a	s	s	p	h	r	a	s	e
4d	79	20	50	61	73	73	70	68	72	61	73	65

4d XOR 61 XOR 68 XOR 65 = 21
79 XOR 73 XOR 72 = 78
20 XOR 73 XOR 61 = 32
50 XOR 70 XOR 73 = 53

Semilla

- La semilla es usada por el PRNG (Pseudorandom number generator) para generar 40 cadenas de 32 bits.
- De estas cadenas se escogerá un bit para generar 4 llaves de 40 bits.

Cifrado (2)

- Dos componentes para cifrar:
  - Algoritmo de chequeo de integridad CRC32: integridad.
  - Algoritmo de cifrado RC4: confidencialidad.
- Proceso:
  - Trama sin cifrar: Hearder+Payload.
  - Calcular el CRC de 32 bits del payload.
  - Añadimos este CRC a la trama.
  - Seleccionamos una llave de 40 bits, de las 4 llaves posibles.

Cifrado (3)

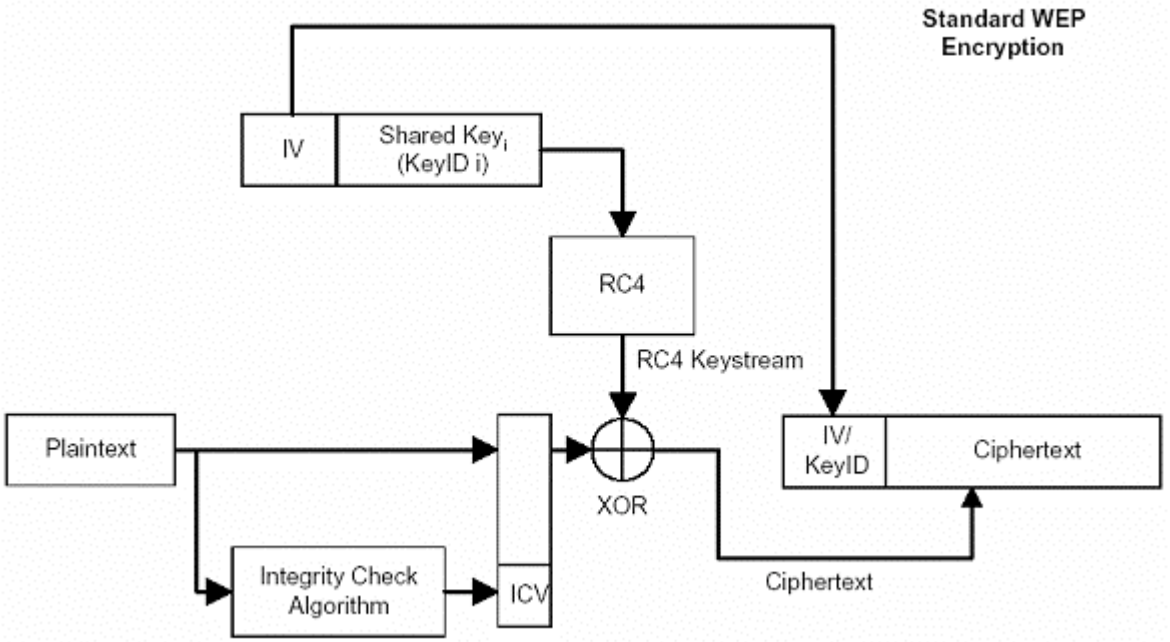
- La misma clave siempre genera el mismo flujo: Agregar a la clave un número aleatorio (IV).
- Añadimos el Vector de Inicialización (IV) de 24 bits al principio de la llave.
- IV: un contador.
- IV de 24 bits y la llave de 40 conseguimos los 64 bits de llave total que utilizaremos para cifrar la trama.
- Aplicamos el algoritmo RC4 al conjunto IV+Key y conseguiremos el keystream o flujo de llave.

Cifrado (4)

- Finalmente añadimos los Headers y el IV+KeyNumber y obtenemos la trama final.
  - IV header=IV(24bits)+Pad(6bits)+KeyID(2bits)

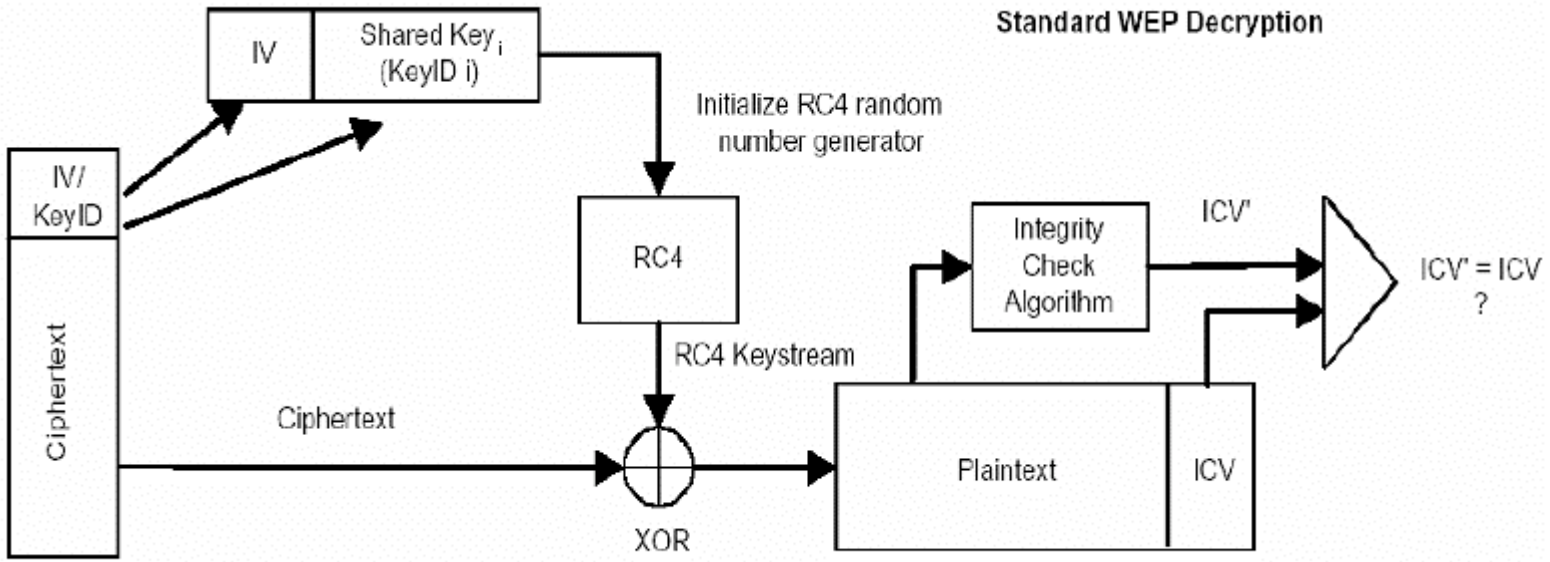
Cifrado (5)

- Esquema cifrado



Cifrado (6)

- Esquema descifrado

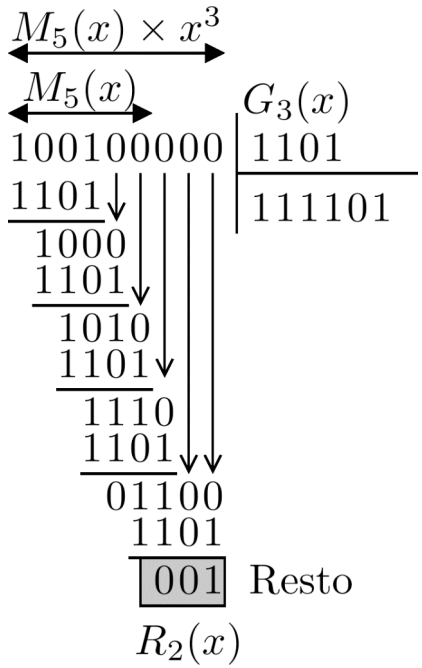


Cifrado (7): CRC-32 (Cyclic Redundancy Check)

- Calculamos CRC de 32 bits de los datos:

  $G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

- Obtenemos un código de integridad ICV.
- Se concatena con la trama. El receptor al descifrar comprobará si la trama se ha alterado.
- Cálculo
  - $M_n(x)$ mensaje grado n
  - $G_k(x)$ polinomio CRC de grado k
  - CRC=$R_{k-1}(x)$ de: $(M_n(x)*x^k)$ mod $G_k(x)$



Cifrado (8): RC4 (Rivest Cipher 4)

- El algoritmo RC4 utiliza claves de 64 bits (40 bits más 24 bits del vector de iniciación IV) o de 128 bits (104 bits más 24 bits del IV).
- Genera un flujo pseudoaleatorio de bits (keystream)
  - Cifrar: XOR entre la keystream y el texto en claro.
  - Descifrar: XOR entre la keystream y el texto cifrado.
- Generación de keystream:
  - S: Permutación de los 256 posibles símbolos de un byte de longitud.
  - 2 índices de 8 bits (i y j).
  - Key scheduling algorithm (KSA): Inicializa la permutación S usando una clave. Desordena una secuencia de números consecutivos.
  - Pseudo-random generation algorithm (PRGA): Genera el flujo de bits cifrados, cambiando la permutación S en cada iteración.

Vulnerabilidades autenticación Shared Key (1)

- Capturamos el 2º y 3er paquete de la autenticación
- 2º paquete tiene el texto del desafío en claro, P
- 3er paquete contiene el desafío encriptado, C
- Conocemos IV
- Se deduce el keystream de tamaño la trama de autenticación:

$$WEP_{PR}^{K,IV} = C \oplus R$$

Vulnerabilidades autenticación Shared Key (2)

- Todos los elementos conocidos y excepto el texto de desafío comunes a todas las Authentication Responses. (número de algoritmo, número de secuencia, status code, element id, longitud, y el texto de desafío)
- Podremos autenticarnos con éxito sin conocer la clave secreta compartida K.
- Calcular ICV válido aprovechando las características lineales de CRC32.

  $CRC(m \oplus k) = CRC(m) \oplus CRC(k)$
- Sólo nos autenticamos, pero todavía no podemos utilizar la red.

Vulnerabilidades cifrado WEP

- Características lineales del CRC
- MIC independiente de la llave
- Tamaño del IV demasiado corto
- Reutilización de IV

Características lineales del CRC

- Nikita Borisov, Ian Goldberg y David Wagner
- ICV: valor para verificar la integridad del mensaje
- El ICV se genera haciendo un CRC de 32 bits, del payload de la trama. Dos problemas:
  - CRCs independientes de la llave utilizada y del IV
  - CRCs lineales: $CRC(m \oplus k) = CRC(m) \oplus CRC(k)$
- Generar un ICV válido haciendo '*bit flipping*' :
  - Modificamos m: $m' = m \oplus \Delta$
  - Calculamos el nuevo ICV válido: $ICV' = CRC(m') = CRC(m \oplus \Delta) = CRC(m) \oplus CRC(\Delta) = ICV \oplus CRC(\Delta)$
  - Tenemos el nuevo paquete cifrado:
    - $k \oplus m' = k \oplus (m \oplus \Delta) = (k \oplus m) \oplus \Delta$
    - $k \oplus ICV' = k \oplus (ICV \oplus CRC(\Delta)) = (k \oplus ICV) \oplus CRC(\Delta)$

MIC independiente de la llave

- David Wagner
- Ausencia de mecanismo de chequeo de integridad del mensaje (MIC) dependiente de la llave.
- MIC que utiliza WEP: simple CRC-32 del payload
- Conocido el plaintext de un sólo paquete cifrado con WEP será posible inyectar paquetes a la red.
  - Capturamos un paquete: $c = m \oplus k$, donde m es conocido
  - Conseguimos parte del keystream $k = c \oplus m$, para el IV concreto
  - Para inyectar m', calculamos: ICV' = CRC32(m') y creamos el nuevo paquete cifrado: $c = (m'|ICV') \oplus k$

Tamaño del IV demasiado corto

- Vector de inicialización (IV): sólo 24 bits de longitud y sin cifrar
- 2^{24} posibles valores de IV
- Red con tráfico intenso: en pocas horas el IV se repetirá

Reutilización de IV

- David Wagner
- WEP no utiliza el algoritmo RC4 “con cuidado” : El IV se repite frecuentemente.
- Ataques estadísticos contra cyphertexts con el mismo IV.
- IV normalmente es un contador que empieza en 0 y se va incrementando de 1 en 1.
  - Reiniciar causa la reutilización de IV’s
  - Tras interceptar suficientes paquetes habrá IV’s repetidos
- Se podrá descifrar ciphertexts interceptados sin conocer la clave.

Ataques

- Ataque de fuerza bruta
- Ataques de diccionario
- Ataques inductivos:
  - Arbaugh
  - Chop-Chop o inductivo inverso
  - Fragmentación
- Debilidades del RC4: Ataques estadísticos

Ataque de fuerza bruta

- Semilla 32 bits a partir de passphrase de caracteres ASCII: Byte más alto de cada carácter 0. Semillas 00:00:00:00 – 7F:7F:7F:7F
- El PRNG es un generador lineal congruente por lo que sólo las semillas de 00:00:00:00 a 00:FF:FF:FF son únicas.
- Semillas 00:00:00:00-00:7F:7F:7F (entropía 21 bits)
- Generando llaves de forma secuencial utilizando las semillas desde 00:00:00:00 hasta 00:7F:7F:7F

Ataque de diccionario

- Utilizar un diccionario para generar sólo las semillas de las palabras (o frases) que aparezcan en el diccionario
- Si la passphrase utilizada está en el diccionario reducimos el tiempo necesario.

Ataques inductivos: Arbaugh (1)

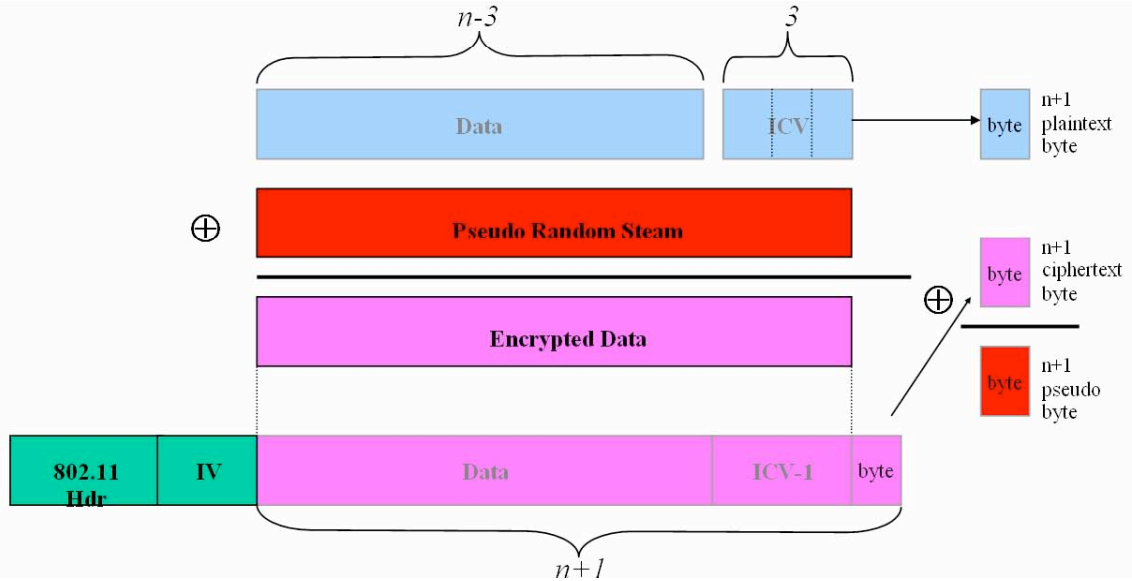
- William A. Arbaugh (2001)
- Se basa en:
  - Características Lineales de CRC.
  - MIC independiente de la llave.
- Conocer parte del plaintext que viaja cifrado en una trama. Ejemplo identificando mensajes "*DHCPDISCOVER*" del que conocemos campos.
- XOR del plaintext conocido con el cyphertext recibido y obtenemos n bytes del keystream para el IV concreto.
- Generamos un paquete ping o ARP Request de tamaño n-3 bytes de longitud.

Ataques inductivos: Arbaugh (2)

FIELD	BYTES	COMMENT
Logical link control	8	Same for all IP packets
IP Header		
Version/header length	1	Same for all IPv4 packets with no options
Differentiated service	1	Default 0 _ 00 common
Length	2	Provided by packet characteristics
Identification	2	0 _ 00 0 _ 00 for many clients
Flags/frag offset	2	0 _ 00 0 _ 00 (no flags, no fragments)
Time to live	1	Commonly used values are 128 and 16
Protocol	1	User datagram protocol (UDP), 0 _ 11
Header checksum	2	Calculable for any guessed header
Source address	4	0.0.0.0 (client yet to have an address)
Destination address	4	255.255.255.255 (broadcast)
UDP header		
Source port	2	DHCP, 68
Destination port	2	DHCP, 67
Length	2	Provided by packet characteristics
Total	34	

Ataques inductivos: Arbaugh (3)

- Calculamos el ICV y añadimos sólo los primeros 3 bytes
- XOR con el keystream y añadimos último byte del ICV para adivinar el siguiente byte del keystream
- Enviamos y esperamos una respuesta probando las 255 posibilidades



Ataques inductivos: Arbaugh (4)

- Repetir hasta tener keystream completo (1500 bytes) válido para un IV.
- Los $2^{24} – 1$ restantes keystreams fáciles de obtener.
- Generar paquete broadcast pings, conocemos plaintext de la respuesta y cada vez con un IV diferente.
- Tabla 2^{24}x1500 = 24GB .

Ataques inductivos: Chop-Chop o inductivo inverso (1)

- Korek (Octubre 2004): herramienta “chopchop”.
- Basado en el ataque inductivo tipo Arbaugh (2001), pero efectuado de forma inversa.
- Aprovecha las propiedades lineales del CRC
  - Parte de un paquete de datos cifrados por WEP .
  - Se decrementa iterativamente byte a byte (con fuerza bruta).
  - Obtenemos el Keystream.
- El AP solo repetirá aquellos paquetes que verifiquen el CRC.
- Tras 256 intentos, se encontrará el byte válido de dicha iteración.
- Herramientas: chopchop, aireplay.

Ataques inductivos: Chop-Chop o inductivo inverso (2)

- Capturamos paquete M válido y creamos paquete M-1 válido

```
_____ DATA ______ ____ICV ___
D0 D1 D2 D3 D4 D5 J3 J2 J1 J0
 +  +  +  +  +  +  +  +  +  +
K0 K1 K2 K3 K4 K5 K6 K7 K8 K9
 =  =  =  =  =  =  =  =  =  =
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9
```

```
_____ DATA ___ ____ICV ___
D0 D1 D2 D3 D4  I3  I2  I1 I0
 +  +  +  +  +  +  +  +  +
K0 K1 K2 K3 K4 K5 K6 K7 K8
 =  =  =  =  =  =  =  =  =
R0 R1 R2 R3 R4 R5 R6 R7 R8
```

  - D0 - D4 igual
  - R5 = I3 + K5 = I3 + (D5+D5) + K5 = (I3+D5) + (D5+K5) = X + S5.
  - R6 - R8 calculados invirtiendo el paso del CRC basándonos en X.
  - Diferencias de I2-I0 y J3-J1 solo dependen de X
  - J0 depende X. K9 = S9 + J0 tendríamos el ultimo byte .

Ataques inductivos: Fragmentación

- Andrea Bittau, Mark Handley y Joshua Lackey
- Conocemos 8 bytes keystream con cabecera
- Mandamos ARPs
- Podemos mandar paquetes con 8 bytes cifrados: 4bytes datos + CRC
- Mandamos 16 fragmentos de 4 bytes y nos devolverán respuesta de 68 bytes: 64bytes datos + CRC
- Mandamos 16 fragmentos de 64 bytes y nos devolverán respuesta de 1028 bytes: 1024bytes datos + CRC
- Con dos fragmentos de ya tendríamos 1500 bytes
- Herramientas: aireplay.

Debilidades del RC4 (1)

- Ataques estadísticos
- Scott Fluhrer, Itsik Mantin y Adi Shamir
- Dos ataques:
  - Patrones invariantes: Si existe en la clave, se propagan al estado interno del algoritmo, debilitándolo.
  - Recuperación de la clave secreta , si clave=IV conocido + clave secreta
- Los bytes iniciales del keystream dependen de tan sólo unos pocos bits de la clave.

Debilidades del RC4 (2)

- Primer byte del keystream es S[S[1] + S[S[1]]]
- 1er byte texto en claro: cabecera LLC (SNAP) para TCP/IP es 0xAA
- IV conocido y es el inicio de la clave. Podemos comenzar RC4 con los 3 primeros bytes de la clave (IV).
- Si S[1] + S[S[1]] ==número byte clave: Podemos deducir el número de la clave si los 3 elementos no cambian
- Probabilidad 5%.

Debilidades del RC4 (3)

- Ataque FMS clásico:
  - Permite derivar la clave (40, 128 bits) una vez capturados unos 8 millones de paquetes de datos cifrados por WEP (con IV únicos):
    - Claves débiles
    - Herramientas: airsnort, wepcrack.
  - Existen firmwares que evitan la generación de IV débiles.
  - Los vectores de interés: [A+3, 255, X], con A índice del byte de la clave a recuperar y X cualquier valor.
  - Solo calcularemos K[A] si sabemos los K[i] anteriores

Debilidades del RC4 (4)

- Implementado por Adam Stubblefield
- 3 primeros bytes conocidos: 0xAA:0xAA:0x03
- Dos métodos:
  - Apuntar paquetes resueltos disminuyendo las posibles combinaciones de bytes de la llave.
  - Supone que el usuario introducirá una llave fácil de recordar que contendrá caracteres ASCII
  - Comprobar si los bytes de la llave son caracteres ASCII (letras o símbolos). Aumentan las posibilidades de adivinar.
  - Con suficientes IVs débiles para un valor concreto de un byte de la llave:
    - Análisis estadístico: tendencia hacia un valor.

Debilidades del RC4 (5)

- KoreK demuestró que los vectores de inicialización descritos en el artículo FMS no son los únicos sobre los que extraer información
- Aircrack comprende 16 tipos de ataques a vectores IV. Son variantes del FMS.

Conclusiones

- WEP no es seguro.
- Solución deficiencias RC4 y WEP: TKIP (Temporal Key Integrity Protocol):
  - Clave combinada MAC y nº secuencia paquete
  - IV de 48bits
  - MIC para evitar ataques inductivos

Líneas futuras

- Vulnerabilidades RC4
- WPA (cifra con RC4), usa TKIP y MIC
  - Debilidades WPA: ataques de diccionario
- WPA2 con AES (Advanced Encryption Standard) en vez de RC4
  - CCMP (Counter Mode with CBC-MAC Protocol) obligatorio
  - TKIP (Temporal Key Integrity Protocol) opcional
- Versiones:
  - Personal autenticación Pre-Shared Key (PSK)
  - Empresarial autenticación 802.1X EAP (Extensible Authentication Protocol)