CORSO DI STUDIO IN INGEGNERIA ELETTRONICA
Laurea Magistrale in Ingegneria delle tecnologie della comunicazione e dell'informazione
"Elementi di crittografia"

DAL WEP AL WPA

Redatto da:
Luca Costantini
Giulio D'Ippolito
Antonio Mariani
Filippo Mazza

Roma, maggio 2010

Anno Accademico 2009/2010

Sommario

Introduzione

Nel seguente elaborato verrà descritta il funzionamento dello standard WEP (Wired Equivalent Privacy) e le sue vulnerabilità che hanno indotto alla successiva diffusione dello standard WAP (Wi-Fi Protected Access).
È stato inoltre descritto nel dettaglio l'algoritmo di cifratura stream RC4, dato il suo utilizzo nel modello di crittografia WEP.
Per completezza di tale lavoro, è riportato nel dettaglio un esempio di attacco ad una rete WPA opportunamente creata (ad hoc), tramite il quale è stato possibile mostrare le vulnerabilità del WEP e la semplicità con la quale è possibile ottenere la chiave di cifratura.

1. WEP (Wired Equivalent Privacy)

Il WEP (Wired Equivalent Privacy) è nato nel lontano 1999 allo scopo di proteggere reti senza fili che per loro natura sono facilmente accessibili. "Wired Equivalent Privacy" significa letteralmente "Privacy come nelle LAN col filo". Purtroppo questo non è più vero; sono infatti state scoperte numerose falle di sicurezza nel protocollo WEP che oggigiorno è quantomeno sconsigliato utilizzarlo.

Il WEP fu il primo standard di crittografia realizzato. In realtà questo è quella parte dello standard IEEE 802.11 che si occupa della sicurezza delle trasmissioni via radio nelle reti wireless.

Esso è infatti il vecchio protocollo standard 802.11 utilizzato per autenticare i partecipanti, che condividono un medesimo segreto (password o chiave privata che dir si voglia), e cifrare conseguentemente le trasmissioni che intercorrono tra access point (AP) e schede di rete wireless degli host in una rete Wi-Fi, a livello quindi data link.

Il WEP fu realizzato con lo scopo di fornire alla reti wireless una sicurezza comparabile con quella delle normali reti LAN (wired) basate su collegamenti via cavo.

Con il trascorrere degli anni seri difetti sono stati riscontrati nell'applicazione di tale protocollo, ed in particolare nell'implementazione dell'algoritmo crittografico utilizzato per rendere sicure le comunicazioni. Infatti allo stato attuale il WEP viene considerato il minimo indispensabile per impedire ad un utente casuale di accedere alla rete locale.

Con il tempo sono stati dunque sviluppati molti attacchi specifici per il WEP; tanto che fu dimostrato che con una semplice analisi del traffico di rete era possibile, nel giro di poche ore, risalire alla chiave di protezione utilizzata; questo a causa della particolare implementazione dell'algoritmo di cifratura stream RC4 (che sarà analizzata nel dettaglio in seguito) utilizzata nel WEP.

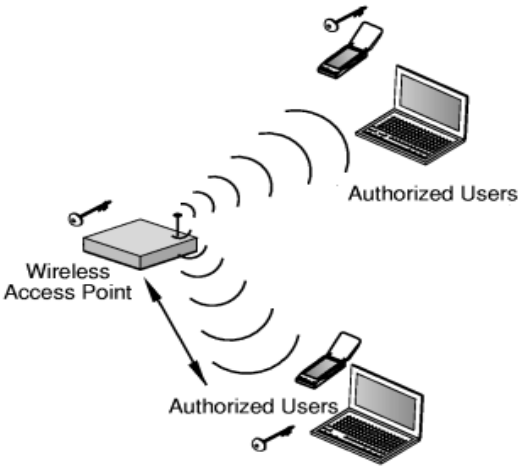
A seguire, come era logico immaginarsi vista la continua evoluzione in questo campo, sono comparsi i primi software in grado di consentire anche all'utente meno esperto di forzare una rete protetta con WEP, sempre nel lasso di tempo di qualche ora.

Ciò premesso, appare evidente come il WEP abbia nettamente fallito il suo modesto obiettivo, infatti la sicurezza ottenibile mediante questo protocollo di sicurezza non è minimamente paragonabile alla garanzia fornita da una rete LAN wired.

Di seguito verranno descritte più nel dettaglio le motivazioni dell'insicurezza del WEP.

❖ INSICUREZZA DEL WEP

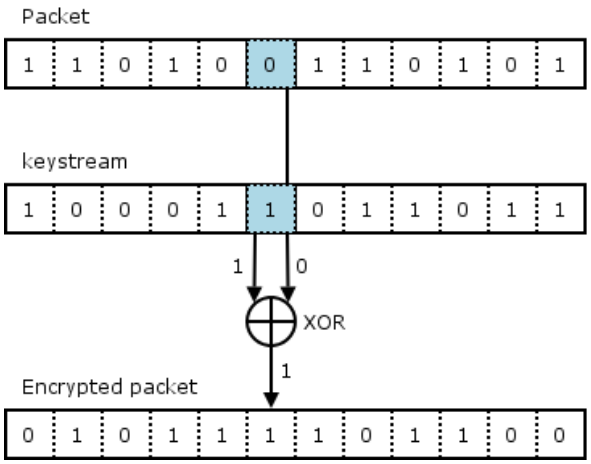
Il WEP ricorre ad una chiave segreta condivisa da tutte le stazioni mobili partecipanti alla rete wi-fi (per stazioni si intendono portatili, server e altri host) e dall'Access Point (o AP). Nelle utenze domestiche spesso l'AP è il modem ADSL Wi-Fi. La chiave segreta è utilizzata per cifrare i pacchetti prima della loro trasmissione nell'etere e un campo CRC aggiuntivo è utilizzato per assicurare che i pacchetti non siano stati modificati durante il loro invio.



La chiave segreta di solito viene impostata a mano sull'AP e sugli host della rete e può essere composta da due chiavi, a 40 bit e a 104 bit. A queste vengono aggiunti 24 bit per il vettore di inizializzazione IV (Initialization Vector) quando viene trasmesso in chiaro.

Per codificare i pacchetti, come detto il WEP si avvale dell'algoritmo di crittografia RC4, un noto cifratore a flusso molto veloce ed efficiente, di cui ne verranno descritte le caratteristiche in seguito.

Un cifratore a flusso non è altro che un PRNG (Pesudo-Random Numers Generetor) cioè un generatore di numeri casuali inizializzato da una specifica chiave segreta: data una chiave il PRNG genera una sequenza virtualmente infinita di bit (il keystream) che viene messa in XOR (⊕) bit a bit con il testo in chiaro. Ovviamente la lunghezza del keystream dipenderà da quanto è lungo il pacchetto da codificare. Il risultato dello XOR è il testo cifrato (pacchetto cifrato) che viene quindi inviato.



XOR		
0	0	0
0	1	1
1	0	1
1	1	0

All'atto pratico il mittente pone in XOR il proprio pacchetto in chiaro M con il keystream K e invia il risultato C.

$$C = M \oplus K$$

Il ricevente, avendo la stessa chiave di inizializzazione per l'RC4, riproduce il medesimo keystream K, lo pone in XOR con il pacchetto cifrato C e ottiene il pacchetto in chiaro M. Cioè, il ricevente esegue:

$$C \oplus K = M$$

Si può notare che il testo ottenuto è nuovamente M, ovvero il testo in chiaro. Ciò è dovuto al fatto che:

poiché $C = M \oplus K$ allora $C \oplus K = (M \oplus K) \oplus K$

Quindi dato che

$$K \oplus K = 0$$

per qualsiasi scelta di K, si ha

$$M \oplus K \oplus K = M \oplus 0$$

Si ottiene pertanto che

$$M \oplus 0 = M$$

perché $M \oplus 0 = M$, per qualsiasi scelta di M.
Gli ultimi due passaggi derivano da proprietà banali dello XOR.
Nelle specifiche della crittografia moderna, si afferma che dato un cifratore a flusso (come ad esempio RC4) è assolutamente vietato il riuso del medesimo keystream per cifrare altri messaggi, in quanto se ad esempio si riutilizza sempre il keystream K per codificare due messaggi M1 e M2, allora si avrebbe che i messaggi cifrati sono dati da:

$$C1 = M1 \oplus K \qquad e \qquad C2 = M2 \oplus K$$

Ponendo quindi in XOR tra loro i messaggi ottenuti (C1 XOR C2), si ottiene lo XOR dei messaggi M1 e M2. La dimostrazione di quanto appena asserito, è riportata di seguito:

$$C1 \oplus C2 = M1 \oplus K \oplus M2 \oplus K = M1 \oplus M2 \oplus \mathbf{K} \oplus \mathbf{K} = M1 \oplus M2 \oplus 0 = M1 \oplus M2$$

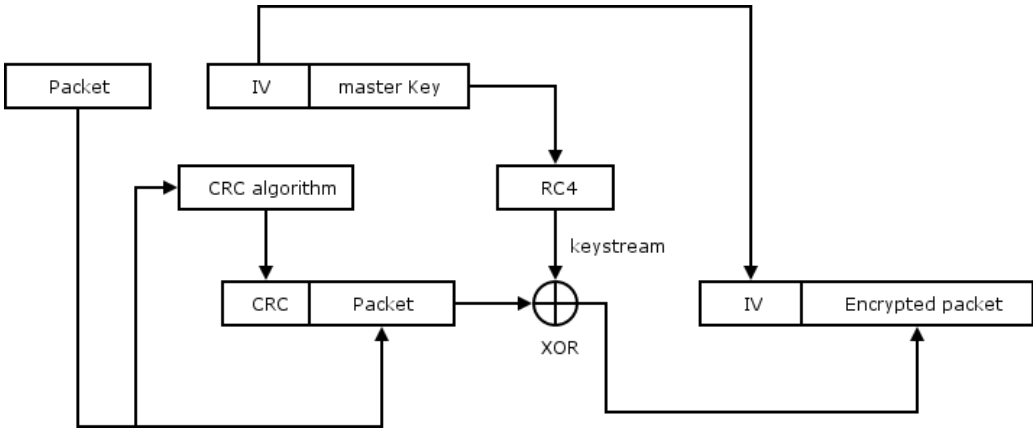
Sapendo che i pacchetti IP hanno spesso gli header abbastanza prevedibili (ad esempio si sa che i 4 byte corrispondenti all'indirizzo IP di destinazione valgono rispettivamente 192.168.1.1, cioè il classico indirizzo locale dell'AP) è possibile ricostruire, seppur parzialmente, i due messaggi M1 e M2, infatti il keystream K si è semplicemente annullato con se stesso dato che

$$\mathbf{K} \oplus \mathbf{K} = 0.$$

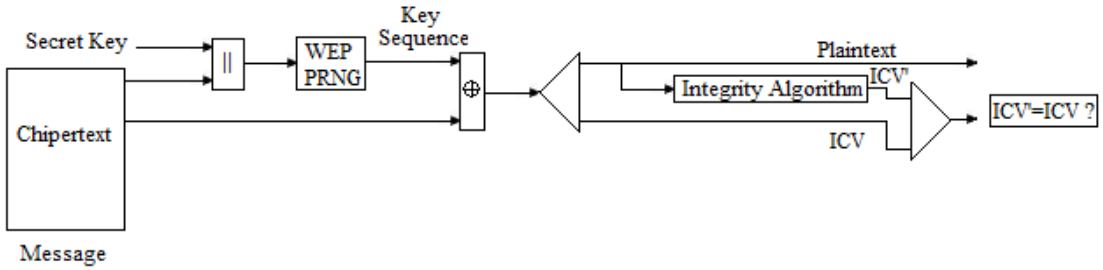
Il WEP rimedia a questo problema utilizzando il vettore di inizializzazione (IV): per evitare il riuso di uno stesso keystream l'IV viene posto in fondo alla chiave segreta condivisa per generare una chiave di inizializzazione per RC4 ogni volta differente. L'IV viene inserito in chiaro nel pacchetto cifrato.

Purtroppo l'IV è un campo di soli 3 byte; ciò implica che possono esserci 2^{24} = 16777216 possibili IV. Potrebbero sembrare molti, in realtà non sono sufficienti a garantire che uno stesso keystream non verrà mai più generato (come invece dovrebbe accadere). Statisticamente infatti, anche scegliendo a caso l'IV per ogni pacchetto e in condizioni di rete non necessariamente a pieno regime, si ha che uno stesso IV si ripete ogni circa 3-4 ore. Ciò significa che circa ogni 3-4 ore si hanno due pacchetti con medesimo IV, cioè codificati con il medesimo keystream! Si potrebbe quindi tentare di decodificare i pacchetti basandoci sulle ricorrenze degli header IP.

Di seguito è mostrato lo schema di funzionamento del WEP al lato mittente:



Come si vede in ingresso all'RC4 entra una chiave creata concatenando un IV, che può essere scelto a caso, in sequenza o in altro modo, e la master key condivisa da tutti partecipanti della rete. L'RC4 genera un keystream che viene messo in XOR bit a bit con il pacchetto originale inclusivo del suo specifico valore CRC. Il risultato è un pacchetto cifrato al quale viene affiancato l'IV assolutamente in chiaro, come si vede dalla figura.
Quanto detto si ripete per ogni singolo pacchetto trasmesso.
Altra nota dolente del WEP è il campo di checksum CRC. Questo campo viene aggiunto dal protocollo WEP per garantire che il pacchetto cifrato non venga manipolato durante il tragitto. Il campo CRC è cifrato insieme al pacchetto vero e proprio.
Il CRC però è generato da un algoritmo di checksum lineare, ciò significa che flippare (scambiare di posizione) l'n-esimo bit nel messaggio comporta solo il dover flippare un set deterministico di bit nel campo CRC per produrre un checksum valido del messaggio modificato.
Un attaccante può quindi flippare tutti i bit che vuole nel messaggio cifrato e poi aggiustare il campo CRC in maniera tale che appaia valido. Va ricordato che il flipping dei bit può essere effattuato anche a codifica RC4 avvenuta, in quanto questo cifratore a flusso lavora bit a bit e usa lo XOR che, per sua natura, è "trasparente" al flipping dei bit.
Per quanto concerne la decodifica, questa segue il seguente schema a blocchi:



Partendo dall'IV ricevuto, lo si aggiunge alla chiave segreta in possesso. Si ottiene così il seme da dare in pasto al PNRG. Si calcola poi lo XOR tra il keystream e il testo cifrato. Si controlla poi, tramite il CRC, l'integrità del pacchetto ottenendo la stringa ICV'(Integrity Check Value) che verrà confrontata con quella implicitamente contenuta nel pacchetto originale (ICV).
Se differiscono si richiede che il pacchetto sia rispedito.

❖ ATTACCHI CLASSICI

Ecco alcuni degli attacchi classici cui è soggetto il WEP:

Decodifica passiva

Deriva dalle considerazioni descritte in precedenza in merito al metodo di codifica con l'operatore XOR.

Un attaccante rimane in ascolto nella rete fintanto di trovare due pacchetti con il medesimo IV. A questo punto gli è sufficiente calcolare lo XOR dei due pacchetti cifrati per ottenere lo XOR dei pacchetti in chiaro (come visto nella dimostrazione precedente). Dallo XOR dei pacchetti in chiaro si può risalire al contenuto di uno dei due (o di entrambi) grazie alla predicibilità dei pacchetti IP che presentano spesso campi uguali, valori ben precisi in determinate posizioni ...

Noto il contenuto di un intero pacchetto si può calcolare molto facilmente il keystream con il quale è stato codificato: infatti, è sufficiente effettuare lo XOR del pacchetto cifrato originale con la versione in chiaro.

$$K = M \oplus C$$

A questo punto qualsiasi altro pacchetto con stesso IV potrà essere decodificato grazie a K (in realtà possono essere recuperati solo i primi N bit del pacchetto se K è lunga N bit)

Active packet injection

Dato che flippare il bit n-esimo del pacchetto cifrato corrisponde a flippare il bit n-esimo della sua versione in chiaro, è possibile manipolare il contenuto di un pacchetto avendo cura ovviamente di modificare opportunamente il campo cifrato CRC flippandone i bit allo stesso modo.

Table-based attack

Lo spazio ridotto dei possibili IV permette di costruire una tabella che mette in corrispondenza gli IV con relativi keystream (keystream recuperati con il primo degli attacchi visti).

Con 2^{24} possibili IV e con una media di 1024 byte a pacchetto si hanno $2^{24} \cdot 2^{10} = 2^{34} =$ 17179869184 byte, cioè all'incirca 16 Gb di spazio richiesto per la tabella. Con questa tabella è poi possibile decodificare qualsiasi altro pacchetto cifrato che transita nella rete.

WEP key scheduling attack

Nel 2001 è stato dimostrato che l'RC4 presenta delle debolezze quando viene usato con chiavi che sono formate dalla concatenazione di una parte segreta ed una parte pubblica esattamente come avviene nel protocollo WEP (master key e IV). In pratica, è sufficiente collezionare un insieme di IV dell'ordine dei 10^5 elementi per risalire alla chiave WEP (la master key). Acquisire 10^5 IV non è un problema dato che la rete è wireless e ogni pacchetto cifrato in transito include un IV in chiaro. Il tempo impiegato può dunque essere più o meno elevato a seconda del carico della rete.

È necessario però precisare che la causa dei problemi di sicurezza del WEP non è data dall'algoritmo RC4, ma da come questo è stato utilizzato ed inserito nel protocollo: l'RC4, se usato correttamente, si è sempre dimostrato estremamente robusto, infatti viene utilizzato nelle cipher-suite SSL senza alcun problema.

Purtroppo ancora oggi molti router e modem ADSL integrano il protocollo WEP e, sebbene spesso se ne possano attivare anche altri (come WPA e WPA2), viene scelto forse per la sua semplicità di configurazione (è sufficiente infatti solo una password) o forse perché di solito é la prima scelta nella lista dei vari protocolli!

A prescindere da ciò, si può affermare che una rete "protetta" da WEP è in realtà assolutamente insicura e apre la strada a tutta una serie di attacchi (eavesdropping in primis) esattamente come una rete in chiaro.

2. RC4

L' **RC4** è uno tra i più famosi e diffusi algoritmi di cifratura a flusso a chiave simmetrica, utilizzato ampiamente in protocolli quali l'SSL (Secure Sockets Layer), per proteggere il traffico Internet, ed il WEP, per proteggere le reti wireless. Nonostante la notevole semplicità e velocità del software, è un algoritmo facilmente violabile ed il suo uso è caldamente sconsigliato se il livello di sicurezza ricercato deve essere elevato.

❖ STORIA

L'RC4 fu sviluppato da Ron Rivest della RSA Security nel 1987: come in altri suoi algoritmi (RC2, RC5 e RC6) la sigla RC sta per **Rivest Cipher** o, alternativamente, **Ron's Code**. Per la sua velocità e semplicità, grazie alla quale è facilmente implementabile sia a livello software che hardware, il suo uso si diffuse speditamente e l'RC4 divenne ben presto l'algoritmo base di importanti protocolli quali il WEP ed il WPA per la cifratura delle comunicazioni nelle schede wireless, e l'SSL ed il TLS per la protezione dei dati delle connessioni internet e delle reti TCP/IP.

L'algoritmo restò inizialmente segreto ma, nel settembre del 1994, comparve, in forma anonima, sulla mailing-list *CypherPunks*. Successivamente, sul newsgroup internazionale *sci.crypt*, fu visualizzato il codice di un algoritmo crittografico i cui risultati erano identici a quelli generati dai programmi che implementavano l'RC4 ufficiale. RSA, nonostante la repentina diffusione di quel codice, non ha mai ammesso che quell'algoritmo fosse l'RC4. Questo ha generato a livello di diritto d'autore una situazione ambigua in cui, nonostante l'algoritmo non sia più segreto, il suo nome è tutt'ora coperto da brevetto. Per evitare quindi possibili ripercussioni legali, generalmente ci si riferisce all'algoritmo chiamandolo **ARCFOUR** o **ARC4**, acronimo di *Alleged RC4* (Alleged in inglese significa *presunto*).

❖ LOGICA DI FUNZIONAMENTO RC4

L'RC4 genera un flusso di bit pseudo-casuali (keystream): tale flusso è combinato mediante un'operazione di XOR con il testo in chiaro per ottenere il testo cifrato. L'operazione di

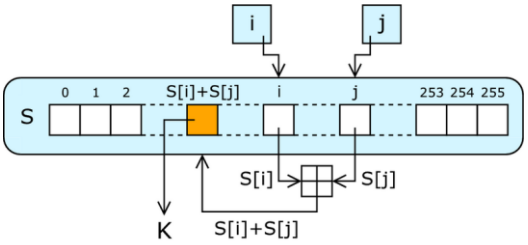
decifratura avviene nella stessa maniera, passando in input il testo cifrato ed ottenendo in output il testo in chiaro (questo perché lo XOR è un'operazione simmetrica).

È costituito da due piccoli algoritmi: **RC4-KSA** e **RC4-PRGA**.

Per generare il keystream, l'algoritmo utilizza una S-box di 256 byte e 2 indici da 8 bit, generalmente identificati con le lettere "i" e "j". La chiave di cifratura fornita dall'utente è generalmente lunga da 40 a 256 bit (da 5 a 32 caratteri) ed è utilizzata per inizializzare l'S-box mediante la funzione *KSA*, acronimo di *Key-Scheduling Algorithm* (cioè "*Algoritmo di gestione della chiave*"). Tale algoritmo può essere rappresentato attraverso il seguente pseudo-codice:

```
for i = 0 to 255
  S[i] = i
next
j = 0
for i = 0 to 255
  j = (j + S[i] + key[i mod keylength]) mod 256
  swap (S[i], S[j])
next
```

dove *key* è la chiave dell'utente, *keylength* è la lunghezza della chiave, *S* è la matrice che rappresenta l'S-box.



Una volta che questo passaggio è completato, il flusso di bit generato utilizzando la funzione di generatore di numeri pseudo-casuali dell'algoritmo (**PRGA**, *Pseudo-Random Generation Algorithm*) è combinato con il testo in chiaro mediante XOR.

Il frammento di codice relativo al PRGA è invece il seguente:

```
i = 0
j = 0
for l = 0 to len(input)
    i = (i + 1) mod 256
    j = (j + S[i]) mod 256
    swap (S[i], S[j])
    output[l] = S[(S[i] + S[j]) mod 256] XOR input[l]
next
```

dove *input* contiene il testo in chiaro, *output* il testo cifrato.

❖ DETTAGLI DI FUNZIONAMENTO

Si parte da un vettore S di 256 elementi i cui valori sono i numeri ordinati da 0 a 255 (ossia il primo elemento è S[0]=0,il secondo è S[1]=1, poi S[2]=2 …S[56]=56…e S[255]=255). Questo è lo STATO INIZIALE che indicheremo con S_0.

Il KSA banalmente ad ogni passo (o step) scambia due elementi di questo vettore S e quindi "lo mescola". La scelta degli elementi da scambiare si basa ovviamente su una chiave fornita in input (per il wep è la perpacked key). Ogni elemento di S viene scambiato con un altro e quindi si verificano 255 SWAP (swap vuol dire appunto scambio). Il vettore dopo il primo swap verrà chiamato S_1, dopo il secondo lo chiameremo S_2 … quindi quello finale lo chiamiamo S_{255}.

Qui il KSA finisce il suo lavoro: in pratica ha solo mescolato un vettore S_0 in base ad una chiave fornendo il vettore S_{255}.

Il PRGA genera il keystream a partire dal vettore S_{255} con alcuni semplici artifici. Tuttavia ai fini di una completa comprensione è meglio fare un esempio che continuare a spiegare a parole.

Di seguito si mostrerà quindi il procedimento mediante il quale creare un keystream a partire da una chiave qualsiasi.

K[0]	K[1]	K[2]	K[3]
23	42	232	11

Per semplicità si sceglie una chiave K a 4 byte, ovvero con 4 caratteri considerando che un carattere è identificato da un numero da 0 a 255 (ad esempio K = 23, 42, 232, 11). Poiché per eseguire l'operazione RC4-KSA è necessario riempire un vettore di 256 elementi, tale chiave K viene ciclicamente ripetuta.

RC4-KSA

Verrà ora descritta la creazione del vettore S_{255}

- *PASSO KSA-0*

Il vettore S viene riempito da 0 a 255 con numeri crescenti. I candidati allo scambio sono:

$$i=0$$

$$j_1 = j_0 + S[0] + K[0] = 0 + 0 + 23 = 23$$

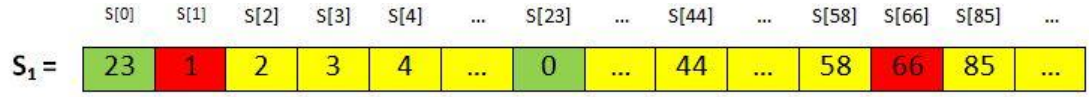
S[0] e S[23] vengono scambiati.

- *PASSO KSA-1*

I nuovi candidati allo scambio sono:

$$i=1$$

$$j_2 = j_1 + S[1] + K[1] = 23 + 1 + 42 = 66$$



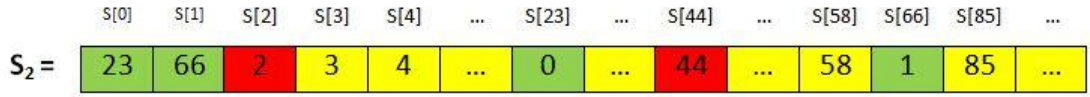
S[1] e S[66] vengono scambiati.

- *PASSO KSA-2*

I nuovi candidati allo scambio sono:

$$i=2$$

$$j_3 = j_2 + S[2] + K[2] = 66 + 2 + 232 = 300 \Rightarrow \text{in modulo 256 è pari} = 44$$



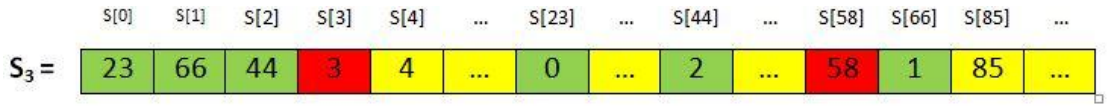
S[2] e S[44] vengono scambiati.

- *PASSO KSA-3*

I nuovi candidati allo scambio sono:

$$i=3$$

$$j_4 = j_3 + S[3] + K[3] = 44 + 3 + 11 = 58$$



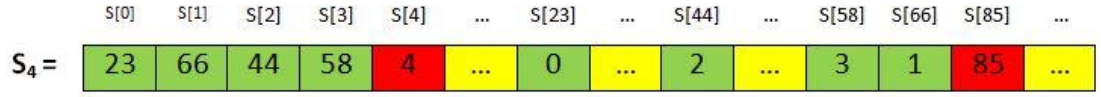
S[3] e S[58] vengono scambiati.

- *PASSO KSA-4*

Non ci sono più elementi nel vettore K quindi ripartiamo da K[0]. I nuovi candidati allo scambio sono:

$$i=4$$

$$j_5 = j_4 + S[4] + K[0] = 58 + 4 + 23 = 85$$



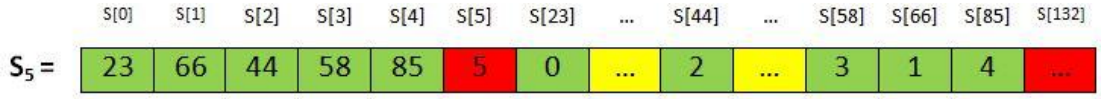
S[4] e S[85] vengono scambiati.

- *PASSO KSA-5*

I nuovi candidati allo scambio sono:

$$i=5$$

$$j_6 = j_5 + S[5] + K[1] = 85 + 5 + 42 = 132$$



E così via per 255 passi!

In questo modo termina l'algoritmo RC4-KSA quando si scambia anche l'ultimo elemento ottenendo S_{255}.

Ottenuto questo vettore, si inizia la parte dell'algoritmo relativa al RC4-PRGA.

RC4-PRGA

Di seguito verrà descritta la creazione del vettore Keystream X

- *PASSO PRGA-0*

Come si nota dallo pseudocodice riportato in precedenza, l'indice i viene inizializzato ad 1 mentre j prosegue con l'ultimo valore del KSA. Quindi i candidati allo scambio nel PRGA sono:

$$i=1$$

$$j_{256} = 12$$

Ecco quindi il vettore creato dalla prima parte del KSA con i candidati allo SWAP evidenziati in rosso:

	S[0]	S[1]	S[2]	S[3]	S[4]	S[12]	S[40]	...	S[56]	...	S[88]	S[141]	S[167]	S[255]
S_{256} =	...	12	28	16	85	248	60	...	151	...	161	134	104	...

Gli elementi del vettore sono aggiornati ed ora si può estrarre il primo valore X[0] del keystream:

$$X[0]= S[\ S[1] + S[12]\ ] = S[260] = S[4] = 85$$

- *PASSO PRGA-1*

I nuovi candidati allo scambio sono:

$$i=2$$

$$j_{257} = j + S[2] = 12+ 28 = 40$$

	S[0]	S[1]	S[2]	S[3]	S[4]	S[12]	S[40]	...	S[56]	...	S[88]	S[141]	S[167]	S[255]
S_{257} =	...	248	28	16	85	12	60	...	151	...	161	134	104	...

Gli elementi del vettore sono aggiornati ed ora si può estrarre il secondo valore X[1] del keystream:

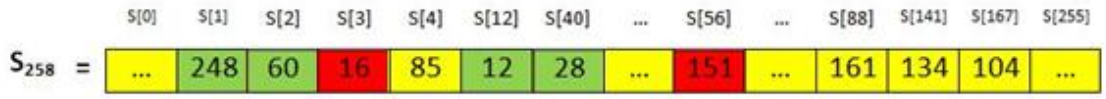
$$X[1]= S[\ S[2] + S[40]\ ] = S[88] = 161$$

- *PASSO PRGA-2*

I nuovi candidati allo scambio sono:

$$i=3$$

$$j_{259} = j + S[3] = 40+ 16 = 56$$



Gli elementi del vettore sono aggiornati ed ora si può estrarre il secondo valore X[2] del keystream:

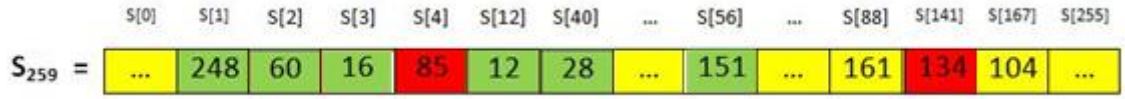
$$X[2]= S[\ S[3] + S[56]\ ] = S[167] = 104$$

- *PASSO PRGA-3*

I nuovi candidati allo scambio sono:

$$i=4$$

$$j_{260} = j + S[4] = 56 + 85 = 141$$



Gli elementi del vettore sono aggiornati ed ora si può estrarre il secondo valore X[3] del keystream:

$$X[2]= S[\ S[4] + S[141] = S[165]$$

Come è possibile notare, si tratta quasi di un algoritmo basato su somme e scambi.

A partire da una chiave di 4 elementi, è stato quindi creato e mescolato con il KSA un vettore per poi, tramite il PRGA, creare i primi 3 byte del keystream X che sono appunto 85, 161 e 104.

Come è già stato descritto nel capitolo precedente, questi byte andranno in XOR con il PAYLOAD ossia con i dati in chiaro.

❖ SICUREZZA

L'RC4 paga la sua semplicità in termini di sicurezza: questa è molto debole e l'algoritmo è violabile con relativa facilità e velocità tanto che il suo uso non è più consigliabile. L'RC4 mostra infatti un comportamento non da vero PRNG: l'analisi del flusso di dati casuali denota una certa periodicità nei primi 256 byte. Ecco perché molte implementazioni dell'algoritmo scartano questi byte facendo eseguire un ciclo di 256 iterazioni a vuoto prima di iniziare ad utilizzare il keystream. Un'altra debolezza è relativa alla forte correlazione che c'è fra la chiave ed il keystream tanto che nel 2005 è stato trovato un modo per violare una connessione wireless protetta con WEP in meno di un minuto.

3. IL WPA COME EVOLUZIONE DEL WEP

Visti i problemi riscontrati nel sistema di cifratura WEP si rese ben presto necessario l'introduzione di un nuovo sistema che colmasse le molte lacune del WEP senza però avere la necessità di un hardware più prestante. In questo modo i possessori di dispositivi con sistema WEP potevano aggiornare, via software, i loro dispositivi senza doverne acquistare di nuovi.
Il meccanismo di standardizzazione del IEEE è un processo molto lungo e complesso. Non essendoci tempo per aspettare la rettificazione dello standard, la *Wi-Fi Alliance* sviluppò un sistema di cifratura alternativo che potesse essere compatibile con i dispositivi che utilizzavano il WEP.
Il sistema che venne sviluppato fu il WPA (Wi-Fi Protected Access). Questo sistema, pur utilizzando sempre l'algoritmo RC4, possedeva notevoli perfezionamenti in termini di sicurezza. I problemi critici del WEP che andavano risolti erano sostanzialmente i seguenti:

- l'autenticazione era mal progettata, in questa fase venivano date indicazioni sulla chiave segreta all'attaccante e inoltre false autenticazioni erano facilmente ottenibili
- la totale assenza di una gestione delle chiavi, la chiave di cifratura veniva utilizzata direttamente all'interno dell'algoritmo RC4 senza mai essere aggiornata.
- gli IV avevano un dominio troppo ristretto: 2^{24} possibili IV non erano poi così tanti, l'invio di due messaggi con medesimo IV comporta un'elevata vulnerabilità.
- l'algoritmo utilizzato per il controllo dell'integrità era lineare, questo permetteva degli attacchi specifici al testo e al controllo che permettevano di validare un testo corrotto.

A tutti questi problemi il WPA riuscì a trovare una buona soluzione.

❖ Generazione delle Pairwise-Master-Key

Il WEP prevedeva l'utilizzo esclusivamente di chiavi Pre-Shared-Key (PSK), il WPA invece possiede due sistemi con cui possono essere generate le chiavi primarie: il primo tramite uno scambio con un server di autenticazione; il secondo basato sul classico sistema a chiave condivisa. Entrambi i sistemi generano una chiave di 256 bit che risulterà:

- la stessa per tutti, nel caso di un sistema a chiave condivisa
- unica per ogni utente nel caso di autenticazione tramite server.

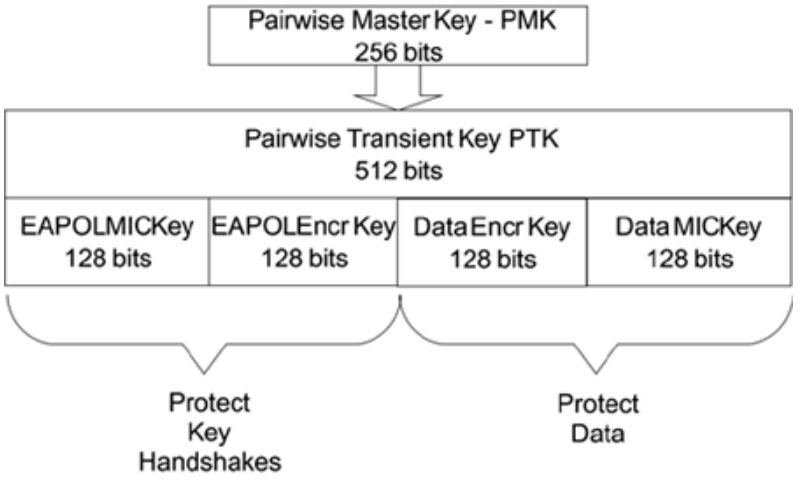
Considerando ora il caso a chiave condivisa, il processo di generazione della chiave è ottenuto elaborando la stringa di caratteri usata come password attraverso il *PBKDF2*, un algoritmo questo che mediante molte iterazioni di funzioni hash produce in output una sequenza di 256 bit data una qualsiasi sequenza di caratteri. In questo modo risulta molto più facile per gli utenti la gestione delle password.
Questa chiave appena generata è la *Pairwise Master Key* (*PMK*).
Totalmente diverso è l'approccio tramite server di autenticazione; l'autenticazione con il server segue le specifiche dell'802.1X.
L'802.1X prevede tre entità il *supplicant*, l'*authenticator* e l'*autentication* server. Queste tre entità rappresentano rispettivamente il dispositivo che si vuole autenticare, l'access point e il server d'autenticazione. In questa fase l'authenticator svolgerà il ruolo di tramite tra le due parti per lo scambio dei messaggi. Supplicant e Autentication Server sono invece a conoscenza di un segreto condiviso che potrà essere posseduto in varie forme (questo grazie all'utilizzo del framework EAP che permette la scelta di diversi sistemi di autenticazione).
Il segreto verrà scambiato tramite protocollo EAPOL (EAP over LAN) tra Supplicant e Autenticator, mentre verrà utilizzato un protocollo di sicurezza specifico tra Authenticator e Autentication Server a seconda del tipo di server di autenticazione utilizzato (RADIUS, Diameter, ecc).
Alla fine di tale processo, qui riportato in maniera molto semplificata, sia il Supplicant che l'Autentication Server avranno generato una coppia di chiavi; di conseguenza l'authentication server invierà questa chiave all'authenticator in modo che entrambi i nodi abbiano la Pairwise Master Key appena generata.
Secondo questo sistema, ogni nodo possiede quindi una chiave che condivide in modo esclusivo con l'access point.

❖ Autenticazione e generazione delle chiavi di cifratura

A prescindere dal modo con cui sono state generate le PMK, la loro gestione e il loro utilizzo rimane comunque lo stesso: la PMK non verrà utilizzata direttamente per cifrare i dati, infatti durante la fase di autenticazione il terminale mobile e l'access point generano tutte le chiavi di cifratura di cui hanno bisogno. Le chiavi, che verranno generate, sono quattro ognuna da 128 bit; nel complesso queste costituiscono la Pairwise Transient Key(PTK) e sono:

- Data Encryption Key
- Data Integrity Key
- EAPOL-Key Encryption key
- EAPOL-Key Integrity key.

Queste chiavi vengono generate tramite un processo di *Four-Way Handshake*, mostrato nella figura a lato, che intercorre tra terminale e access point.



Esse verranno rinnovate periodicamente o ogni qual volta che un terminale effettua l'accesso.

Questo processo produce due importanti risultati: il primo è la generazione delle chiavi di sessione che verranno poi effettivamente utilizzate per cifrare, il secondo, non meno importante, è di autenticare mutuamente le due parti.

È necessario precisare che gli attributi che vengono elaborati per la generazione delle chiavi sono:

- la PTK,
- gli indirizzi MAC dei due dispositivi
- due sequenze generate casualmente, una per terminale (chiamate *nonce*).

Il four way handshake garantisce che entrambi i dispositivi generino le medesime chiavi senza che sia possibile per un attaccante in ascolto ottenere informazioni su di esse.

Il processo con cui vengono generate è il seguente: Supplicant e Authenticator sono entrambe a conoscenza della PTK, l'Authenticator invia dunque un messaggio contenente esclusivamente l'*Anonce* (nonce generata dall'Authenticator) in chiaro. A questo punto il Supplicant si genera la sua *Snonce* ed è in grado di generarsi le quattro chiavi di cifratura che occorreranno nei vari processi. Il messaggio di risposta sarà costituito dal Snonce in chiaro con in coda il MIC (codice di controllo dell'integrità che verrà descritto in seguito) cifrato con la chiave EAPOL-Key Integrity Key. Dopo questa fase anche l'Authenticator ha ottenuto le chiavi di cifratura ed è inoltre in grado di verificare se il MIC è stato generato con la chiave corretta; questo è sufficiente come prova della sua autenticità.

Nella fase successiva, l'Authenticator dà conferma al Supplicant inserendo anche esso un MIC generato con la chiave EAPOL-Key Integrity Key. La risposta del Supplicant conclude il processo di four-way hadshake e consente l'inizio delle trasmissioni cifrate.
Si precisa che, quanto appena descritto, fa riferimento a scenari di trasmissione unicast; nel caso che l'access point voglia inviare a più utenti alcuni pacchetti, quest'ultimo si genererà una *Group Master Key* (se non ne possedesse già una) da cui calcolerà la chiave di sessione *Group Transient Key* (*GTK*) di 256 bit che viene poi spezzata in due chiavi: la *Group Encryption Key* e la *Group Integrity Key* in fase di utilizzo. La GTK verrà trasmessa dall'access point singolarmente ad ogni terminale in modo sicuro.

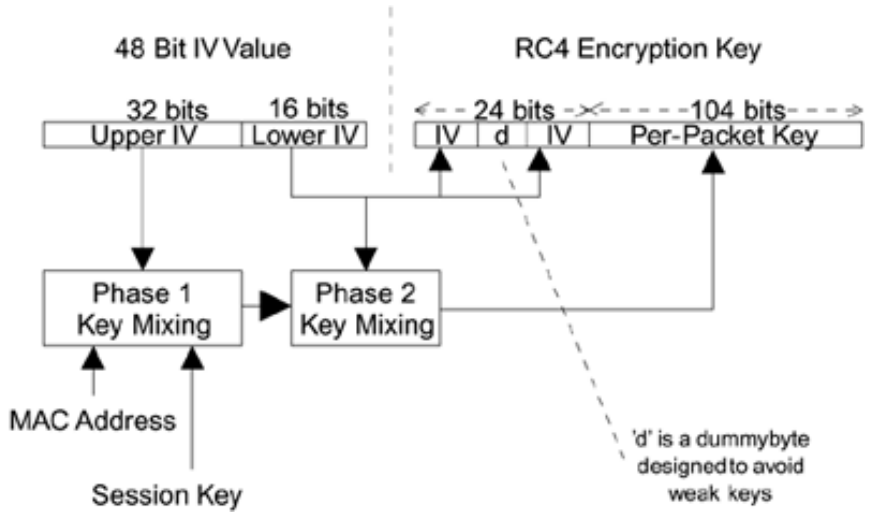
❖ TKIP (Temporal Key Integrity Protocol)

Questo algoritmo è stato generato dovendo tener conto della necessità di poter introdurre questo sistema all'interno dei dispositivi che utilizzavano il WEP. Per garantire maggiore efficacia si è pensato di generare una chiave diversa a ogni trasmissione attraverso un'elaborazione della chiave che conferiva maggiore robustezza. Questo processo doveva tener conto del fatto che il WEP utilizza l'RC4 elaborando un IV di 24 bit; tale sequenza presentava però alcuni problemi che dovevano essere risolti:

- la sequenza era troppo piccola: 2^{24} possibili valori possono esaurire nell'arco di qualche ora,
- la possibilità che uno stesso IV fosse generato in dispositivi diversi non era in alcun modo evitato,
- non incorporava un contatore per evitare i replay attack,
- molti IV generavano delle chiavi deboli che indebolivano notevolmente il sistema.

A tal proposito, per risolvere il primo e l'ultimo problema, si decise di aggiungere 32 bit all'IV ottenendo una sequenza di 56 bit, a cui però vengono sottratti 8 bit che contenevano le chiavi deboli. L'IV utilizzato sarà quindi di 48 bit.
Il replay attack venne risolto dando il valore di contatore ad una porzione dell'IV.
Questa sequenza deve però essere riportata a 24 bit per essere compatibile con i sistemi esistenti e inoltre bisognava evitare il più possibile l'utilizzo dello stesso IV da parte di dispostivi diversi. L'idea fu di spezzare pertanto l'IV in due parti, la prima da 32 bit la seconda da 16 bit, ed effettuare due elaborazioni successive. La prima parte dell'IV aveva inoltre valore di contatore e veniva incrementata unitariamente ad ogni trasmissione.

La fase iniziale era costituita dai 32 bit dell'IV che venivano elaborati insieme all'indirizzo MAC e alla chiave di sessione. Questa sequenza così generata costituisce la chiave che verrà passata all'algoritmo RC4 insieme a quei 16 bit dell'IV (gli 8 bit mancanti sono mantenuti fissi per evitare chiavi deboli).
Come si può notare ogni pacchetto sarà cifrato con una chiave diversa, l'estrazione di questa chiave non è ottenibile senza possedere la chiave di sessione.
In fase di decifratura il ricevente sarà il solo in grado di calcolarsi la chiave utilizzata per cifrare.



❖ MIC (Message Integrity Code)

Nel WEP il controllo di integrità veniva concatenato ad ogni pacchetto da cifrare; tale controllo veniva eseguito con un algoritmo lineare: il CRC-32. L'uso di questo sistema, come già accennato in precedenza, non garantiva robustezza alla manomissione volontaria dei pacchetti. L'approccio utilizzato nello standard WPA fu leggermente differente; in esso si usa infatti il MIC per applicare il controllo di integrità.
Il MIC è costituito da una sequenza di 64 bit ottenuta elaborando un intera MAC Service Data Unit (MSDU), intestazioni MAC incluse. La sequenza è inoltre cifrata utilizzando la Data Integrity Key che, essendo di 128 bit, viene divisa in due parti da 64-bit che verranno usate distintamente dalle due entità preposte a comunicare.

Questo campo risulta essere molto suscettibile ad attacchi tanto che nel caso venga individuato un possibile attacco volontario all'integrità il sistema mette in piedi delle contromisure atte ad alleviare l'efficacia di questi attacchi.

❖ Osservazioni

Come visto, il WPA ha raggiunto un livello di sicurezza piuttosto accettabile, obbiettivo che si presentava piuttosto difficile da ottenere visti i vincoli realizzativi presenti. La compatibilità con i sistemi già presenti nel mercato ha inoltre ridotto notevolmente le possibilità di raffinare il sistema.
È importante però fare un oculata scelta della chiave: un attacco diretto al dizionario potrebbe generare dei buoni risultati.

4. ESEMPIO PRATICO DI ATTACCO A RETE WEP

In questa sezione si presenta un esempio di attacco alla rete WEP mediante software reperibile facilmente online. L'esempio non vuole essere un incoraggiamento a tali attività che, se non autorizzate, possono avere risvolti legali.
Il sistema operativo usato è una distribuzione Linux ed i comandi indicati saranno intesi come eseguiti al terminale (riga di comando Shell o simile) secondo la convenzione

```
#comando
```

Prima di procedere si analizza la procedura con cui due dispositivi entrano in comunicazione.
Le fasi possono essere così schematizzate:

1. la station ricerca eventuali access points (AP) nel raggio di copertura ed ascolta i messaggi provenienti; impostato nella Station un SSID particolare, questa cercherà messaggi provenienti all'AP con quel SSID;
2. per collegarsi, invierà una richiesta di autenticazione all'AP scelto;
3. l'AP autentica la station; a riguardo la WEP fornisce due tipi di autenticazione:
    a. *Sistema Aperto* (Open System), dove chiunque può accedere alla rete e ricevere qualunque messaggio non crittografato; è la modalità di default del protocollo 802.11.
    b. *Chiave Condivisa* (Shared Key), in cui solamente i clients a conoscenza della chiave di autenticazione possono accedere alla rete.
4. la station richiede l'associazione all'AP;
5. l'AP e la station si associano; la station è ora registrata;
6. si può procedere alla comunicazione dei dati.

Verrà ora analizzata meglio la fase di autenticazione a chiave condivisa, che suscita interesse in quanto prevede che AP e client abbiano la stessa chiave per poter procedere nella comunicazione.
Dopo l'invio della richiesta di autenticazione, da parte della station, l'Access Point invia del testo di prova alla station, la quale crittografa il testo con la chiave in suo possesso e lo rimanda all'AP. Questi decritta il testo ricevuto e, in caso di corrispondenza con quello che aveva

inviato, autentica il client. Qualora non vi sia corrispondenza fra testo inviato e decrittato, l'autenticazione verrà negata. La chiave WEP dunque non è solo usata per crittografare la parte dei messaggi contenente i dati, come nell'autenticazione a sistema aperto, ma anche nella fase di autenticazione.

Questa chiave, come detto nei capitoli precedenti, verrà combinata con i vettori di inizializzazione (IVs) per produrre il flusso di chiave, o keystream. La chiave di 40 o 104 bits (5 o 13 caratteri ASCII, questa ultima nota come WEP2 [7]) è concatenata ai 24 bit dell'IV ed il tutto passa all'algoritmo dell'RC4 che restituisce il keystream.

Per ottenere il messaggio crittografato, il messaggio in chiaro, concatenato al suo controllo di integrità, verrà messo in XOR con il flusso ottenuto.

Il punto debole del sistema, che si sfrutta in questa sezione per l'attacco, è relativo agli IV: il protocollo 802.11 permette che pacchetti successivi possano utilizzare IV identici. Essendo inoltre il campo, relativo all'IV, di 3 byte è possibile, creando un opportuno traffico nella rete, far sì che questo si ripeta con maggiore frequenza di quanto non accadrebbe in condizioni nominali.

Nota la presenza in chiaro degli IV nei pacchetti è possibile passare alla fase operativa: un esempio di attacco a chiave WEP.

Non è difficile reperire online programmi molto semplici da usare per effettuare questo tipo di attacchi; nei sistemi linux ad esempio è già presente nei repository un programma apposito: *aircrack* [2]. Questa utility, composta di più applicazioni, permette di recuperare la chiave una volta raccolti un numero sufficiente di frame inviati nella rete. Per fare ciò occorre monitorare la rete wireless con l'interfaccia radio, detta comunemente scheda di rete senza fili.

La modalità della scheda che indica l'uso per l'ascolto passivo dell'etere prende appunto il nome di *modalità monitor*.

Non tutte le schede wireless e non tutti i driver relativi permettono questa modalità; su internet si possono trovare liste di schede e driver compatibili, nonché relative modifiche agli stessi.

Avendo una di queste schede con i relativi driver nei sistemi Linux, impostare la modalità monitor è molto semplice grazie ai comandi forniti da *iwconfig*, utility specifica per le interfacce wireless usata anche per impostare frequenza, potenza e modulazione.

Con *iwconfig* è sufficiente digitare

```
#iwconfig <interfaccia> mode monitor
```

Esiste una utility compresa nel programma *aircrack* che imposta semplicemente delle interfacce virtuali denominate *"mon"* da cui monitorare la rete; l'utility indica anche i programmi che

usano, o potrebbero usare, la scheda di rete fisica e dare problemi durante le operazioni di recupero della chiave. Con questa utility si ottiene il beneficio che con la scheda reale si può trasmettere mentre l'interfaccia virtuale è in monitor. Il comando è il seguente:

```
#airmon-ng start <interfaccia> <canale wireless>
```

Il canale da indicare è quello su cui è attiva la rete; per scoprire quale sia, basta fare una scansione con *iwlist*, altra utility linux che visualizza le reti senza fili presenti, con i parametri che le riguardano come potenza del segnale, velocità, canale, ecc. Il comando è:

```
#iwlist <interfaccia> scan
```

Attivata la modalità di ascolto suddetta, si procede all'analisi delle reti con l'utility *airodump-ng*, anch'essa compresa nel programma *aircrack*. Sempre da terminale il comando è:

```
#airodump-ng <interfaccia>
```

L'obiettivo è di catturare pacchetti fra due dispositivi che conoscono la chiave in modo da collezionare vettori di inizializzazione da analizzare in un secondo momento.
Il numero di pacchetti necessari non è noto a priori, dipende da più fattori, come il riuso degli IV e la lunghezza della chiave. Si precisa però che la lunghezza della chiave non è mai indicata dall'AP, neanche nei pacchetti che lancia per gestire la rete.
Se non sono specificate opzioni, con il comando precedente è possibile mostrare al terminale le reti trovate su tutti i canali della banda dei 2.4 GHz. È altresì possibile indicare il canale di interesse o la modalità del protocollo anche sulla banda dei 5 GHz con schede che supportano queste frequenze (modalità a/b/g). Per controllare le frequenze supportate, è sufficiente l'utility *iwlist* suddetta, digitando a terminale

```
#iwlist <interfaccia> frequency
```

Trovare il canale di ascolto (qui indicato con X) e fissarlo, rende migliori le performances, poiché si perderà meno tempo nell'ascolto dei canali che non interessano: durante l'ascolto degli altri canali si perderebbero i pacchetti sul canale d'interesse. Ciò può essere messo in pratica attraverso il comando seguente:

```
#airodump-ng -c X
```

Il programma *airodump-ng* scansionerà la banda alla ricerca di Access Points e Clients, indicando MAC, potenza del segnale, numero di pacchetti catturati, canale, crittografia usata e nome della rete con Mbps (se si tratta di AP).
Per concentrarsi su una rete in particolare, è possibile indicare l'ESSID od il BSSID, come nel caso che si sta considerando, dato che la rete è di tipo ad hoc (il BSSID è generato in maniera random alla creazione della rete ed è composto di 48 bit). Pertanto si digita il comando:

```
#airodump-ng -d <SSID>
```

Per far sì che *airodump* scriva su un file i dati estrapolati dai pacchetti, in particolare gli IV, occorre indicare l'opzione `-w`, seguita dal prefisso che sarà inserito nel nome del file.
Per salvare solo gli IV c'è l'opzione `-i`.
Il file contenente i dati sarà creato nella cartella da cui si lancia l'utility.
Per terminare l'operazione è necessario premere la combinazione di tasti "`ctrl-c`".
Come detto, per impostare la rete di interesse, qualora ve ne siano più di una, si utilizza l'opzione `-d <BSSID>`.
Qualora non vi fossero utenti collegati alla rete, il traffico generato sarebbe composto solo dai i beacon inviati dall'AP, trasmissioni queste che forniscono le informazioni base della rete a chiunque (SSID, canale, crittografia usata, ...).
Ai fini di tale elaborato, il traffico utile per comunicare è generato dagli utenti che scambiano pacchetti con l'AP. Questo traffico come detto potrebbe essere esiguo per avere una raccolta necessaria a craccare la WEP in breve tempo; per poter generare traffico è possibile iniettare frames nella rete e monitorare le risposte dei dispositivi.
Fuori dal contesto WEP, inserzioni di richieste specifiche possono inoltre causare disassociazioni fra utenti ed AP, utili per catturare l'handshake WPA che un utente potrebbe generare se eventualmente tentasse di ricollegarsi.
Inoltre gli access point con nomi nascosti (non resi pubblici e dunque non rilevabili con le utility classiche) potrebbero essere scoperti se si inviano richieste di de-autenticazione ai clients.
Non tutti i drivers permettono queste funzioni ed alcuni necessitano di patches per poter monitorare il traffico ed al tempo stesso fare packet-injection.
Alcune schede dovranno utilizzare drivers particolari o modificati come quelli madwifi.
Per verificare che la modalità *iniezione di pacchetti* funzioni, si può lanciare il comando

```
#aireplay-ng <interfaccia in monitor> -e <SSID>
```

che indicherà anche le statistiche del funzionamento. Un tipo di iniezione possibile è quella di richiedere la de-autenticazione immedesimandosi in uno dei clients già associati.
Dato che questa operazione fa cancellare al dispositivo la tabella degli indirizzi fisici, una nuova comunicazione obbligherà il client a riaggiungere l'indirizzo fisico dell'AP nella tabella tramite protocollo ARP. Molti dispositivi si ricollegano in automatico; riconoscere le richieste ARP per reiniettarle non è una operazione complicata.
Uno dei punti di forza dei sistemi linux, scelti per la dimostrazione, è proprio nella possibilità di operare facilmente in maniera nativa sugli strati più bassi dei protocolli, ad esempio agendo sullo strato MAC per camuffare l'indirizzo fisico ed immedesimarsi in altre macchine.

```
#aireplay-ng --arpreplay
#aireplay-ng
--deauth=<tentativi>
-b <MAC dell'AP> ?????
-h <MAC del Client associato>
<interfaccia>
```

L'importanza della scelta di un client già associato (inserendo l'indirizzo MAC di quel terminale) consiste nella ricerca di nuovi IV, che non verrebbero ad esserci altrimenti, poiché l'AP scarterebbe quei pacchetti.
Terminata la raccolta di dati si passa il materiale al programma di decrittazione vero e proprio, *aircrack*:

```
#aircrack-ng <file .ivs>
```

❖ L'ATTACCO

Analizziamo ora come funziona l'attacco. Le modalità possibili sono molteplici, in quanto, con gli anni, varie strategie sono state implementate, come quella proposta nel 2001 da Fluhrer, Mantin e Shamir, e quella del 2004 proposta da un anonimo sotto pseudonimo "KoreK", il quale realizzò anche un algoritmo pratico.

La prima versione del programma utilizzato nasce proprio in quegli anni ed è andato evolvendosi col tempo. Nel 2005 Andreas Klein trovò nuove correlazioni fra keystream e chiave rispetto alle precedenti analisi. La quantità di dati e tempo richiesti variano sensibilmente a seconda della modalità scelta nell'attacco.
Infine, nel 2007, viene pubblicata la strategia di Erik Tews, Ralf-Philipp Weinmann e Andrei Pyshkin, strategia questa di default applicata nel programma air crack [15], usato in questo progetto per la realizzazione dell'attacco.
Poiché sono possibili combinazioni delle tecniche e bruteforce, le quali spesso sono anche impostate come opzione di default, verranno ora analizzate alcune eventuali strategie.

Attacco FMS

L'attacco di Fluhrer, Mantin e Shamir (FMS) si basa sulle seguenti considerazioni: dato l'IV in chiaro nel pacchetto, un attaccante conosce i tre bytes ad esso relativi che vengono concatenati alla chiave usata ed inseriti nell'algoritmo RC4. Conoscendoli, si può risalire ai primi tre cicli dell'algoritmo.
Il protocollo 802.11 non indica come generare questi IV; pertanto i produttori, nella pratica, hanno implementato due modalità: prendere una distribuzione uniforme o andare in ordine crescente.
Per riuscire a trovare il byte successivo al vettore di inizializzazione occorre fare una osservazione: i primi bytes del testo in chiaro dei pacchetti inviati sono piuttosto semplici da indovinare dato che contengono le informazioni che i protocolli richiedono per la comunicazione, come la versione del protocollo, il tipo di richiesta, la presenza o meno di crittografia WEP, indirizzi MAC e simili. Ciò è facilitato dalla possibilità di inviare richieste arbitrarie alla rete anche senza chiave, generando traffico. Poiché i pacchetti sono messi in XOR con il keystream è facile risalire con queste supposizioni ai primi bytes di quest'ultimo.
Se alcune condizioni (*resolved conditions*) sono soddisfatte, è possibile prevedere il comportamento di alcuni passi dell'RC4 e calcolare il byte successivo.
Un attaccante potrebbe dunque collezionare IV, e le rispettive parole crittografate, sceglierne un sottoinsieme, per cui queste condizioni sono verificate, generare il flusso e risalire con una certa probabilità al byte seguente; la procedura può essere poi reiterata. I possibili valori che quel byte può avere, vengono "votati", così da poter tornare indietro nella procedura se la decodifica non è andata a buon fine e scegliere un valore con voto simile ma non "eletto".

Il calcolo delle probabilità indica che seppure quelle condizioni si verificano per un numero esiguo di IV, è possibile, con circa 5 milioni di pacchetti, avere una probabilità del 50% di recuperare la chiave.
Dopo la pubblicazione dell'articolo però, i produttori dei dispositivi iniziarono a rilasciare patches che evitavano l'uso degli IV per i quali era semplice l'attacco, limitandone gli effetti.

Attacco Chopchop

Successivamente KoreK trovò nuove correlazioni nelle fasi dell'RC4 e propose una implementazione che utilizza 17 differenti attacchi. Nella pratica il più semplice per recuperare il keystream è quello denominato *Chopchop*.
Questo si basa sull'assunzione di avere un "oracolo" che dica se il checksum di un pacchetto crittografato è corretto o sbagliato. KoreK constatò che è possibile risalire agli ultimi m bytes del keystream usato, il che implica una media di 128 m richieste all'oracolo; l'autore mostrò inoltre che un access point poteva essere usato proprio come oracolo.
Erik Tews nel suo lavoro "Attacks on the WEP Protocol" notò che vi sono almeno due metodi per fare ciò:

- il primo, in cui l'attaccante inietta un pacchetto per un host e controlla se l'AP rilancia o meno il messaggio;
- il secondo, l'attaccante invia un messaggio come stazione non collegata alla rete e vede se l'AP indica o meno di collegarsi prima di trasmettere.

Questo attacco è molto efficace non solo per la semplicità, ma anche perché pacchetti con checksum errato vengono scartati allo strato fisico, e non viene notificato nulla agli strati superiori, dove risiedono solitamente i sistemi di sicurezza.

❖ ESECUZIONE DI AIRCRACK

Si prosegue ora spiegando come procede il programma; per far ciò è stato analizzato il codice sorgente, disponibile su internet presso [2].
Il programma accede al file nel quale è stato salvato l'output di *airodump-ng* e, dopo aver saltato i frame di controllo (inutili per lo scopo di tale progetto), scorre fra i pacchetti cercando richieste e risposte di associazione. Un thread calcola un voto su un sottogruppo di IV e

successivamente partono gli attacchi; di default la tecnica usata è quella proposta da Andrei Pyshkin, Erik Tews e Ralf-Philipp Weinmann, ma è possibile anche adottare la strategia di Korek. Fra le possibilità vi è anche una modalità sperimentale di attacco a forza bruta (bruteforce).

L'esito della ricerca della chiave WEP dipenderà dalla quantità di vettori di inizializzazione raccolti. Un fattore da sottolineare è la possibilità di attivare un bruteforce sugli ultimi bytes della chiave quando i precedenti sono stati trovati, così da velocizzare le operazioni.

Per una WEP di 5 caratteri nei test effettuati sono bastati circa cinquemila IV, mentre oltre sessantamila non sono stati sufficienti per una WEP di 13 caratteri.

Secondo la documentazione del programma, con l'attacco di default (PTW) servono circa 20000 pacchetti per chiavi a 64 bit e fino a 85000 pacchetti per la versione a 128 bit [8].

Il tempo di elaborazione, con un computer portatile di basse prestazioni, non ha superato i cinque minuti.

L'esempio mostrato espone chiaramente quanto è vulnerabile una rete con crittografia WEP.

Alcuni studi [5], [6] già vari anni fa sottolineavano che, nonostante il rischio, poco meno della metà degli AP usava questa crittografia. Eppure, ancora oggi molti dispositivi usano WEP, sebbene nuovi protocolli (WPA2, ecc..) stanno prendendo il suo posto.

Bibliografia

[1] Wireless Networking Basics, NETGEAR, Inc., Ottobre 2005

[2] http://www.aircrack-ng.org

[3] http://www.hsc.fr/, Hervé Schauer Consultants, Network Security Consulting Agency Since 1989

[4] Wi-Fi security – WEP, WPA and WPA2, Guillaume Lehembre, Giugno 2005, www.hakin9.org

[5] Exploiting and Protecting 802.11b Wireless Networks, Craig Ellison, 2001, http://www.extremetech.com/article2/0,3973,9280,00.asp

[6] Wireless Security Threat Taxonomy, Donald Welch and Scott Lathrop, Proceedings of the 2003 IEEE Workshop on Information Assurance United States Military Academy, West Point

[7] WEP2 Security Analysis, Bernard Aboba, Microsoft, May 2001, https://mentor.ieee.org/802.11/dcn/01/11-01-0253-00-000i-wep-2-secureity-analysis.ppt

[8] http://www.aircrack-ng.org/doku.php?id=simple_wep_crack

[9] Attacks on the WEP protocol by Erik Tews, December 15, 2007

[10] http://www.cdc.informatik.tu-darmstadt.de/aircrack-ptw/

[11] http://www.netstumbler.org/f49/need-security-pointers-11869/ (2004)

[12] Weaknesses in the Key Scheduling Algorithm of RC4, Flurer et al., 2001

[13] http://www.linuxjournal.com/article/8312

[14] http://blogs.zdnet.com/Ou/index.php?p=20

[15] Breaking 104 bit WEP in less than 60 seconds, Erik Tews, Ralf-Philipp Weinmann, and Andrei Pyshkin, FB Informatik, Germany, 2007