Wireless Network Esercitazioni

Alessandro Villani
avillani@science.unitn.it

WEP Cracking

WEP: Wired Equivalent Privacy

- L'obiettivo dichiarato della chiave WEP (Wired Equivalent Privacy) è (*Nomina sunt consequentia rerum*) quello di fornire sul canale wireless un livello di sicurezza equivalente a quello che ci si può aspettare nel caso di reti wired
- Alcuni l'hanno pensato come unico meccanismo per il controllo di accesso
- Altri come soluzione dei problemi di sicurezza

WEP: Wired Equivalent Privacy

- La chiave condivisa deve essere installata sugli Access Point e sui client
- Sugli apparati sono configurabili fino a 4 chiavi ma lo standard non specifica come queste chiavi vanno gestite: nella pratica ne viene usata una sola
- PROBLEMA:
  - Non è possibile installarla/aggiornarla automaticamente
  - È uguale per tutti gli utenti dello stesso AP

WEP: Come Funziona

- WEP si basa sull'algoritmo RC4 della RSA
- È un sistema di crittazione basato su una chiave condivisa
- La chiave condivisa è lunga 40 bit (o 104 bit)
- È concatenata a un ***vettore di inizializzazione*** (IV) lungo 24 bit
- Si ottiene cosi un seed di 64 bit (o 144 bit) per l'RC4

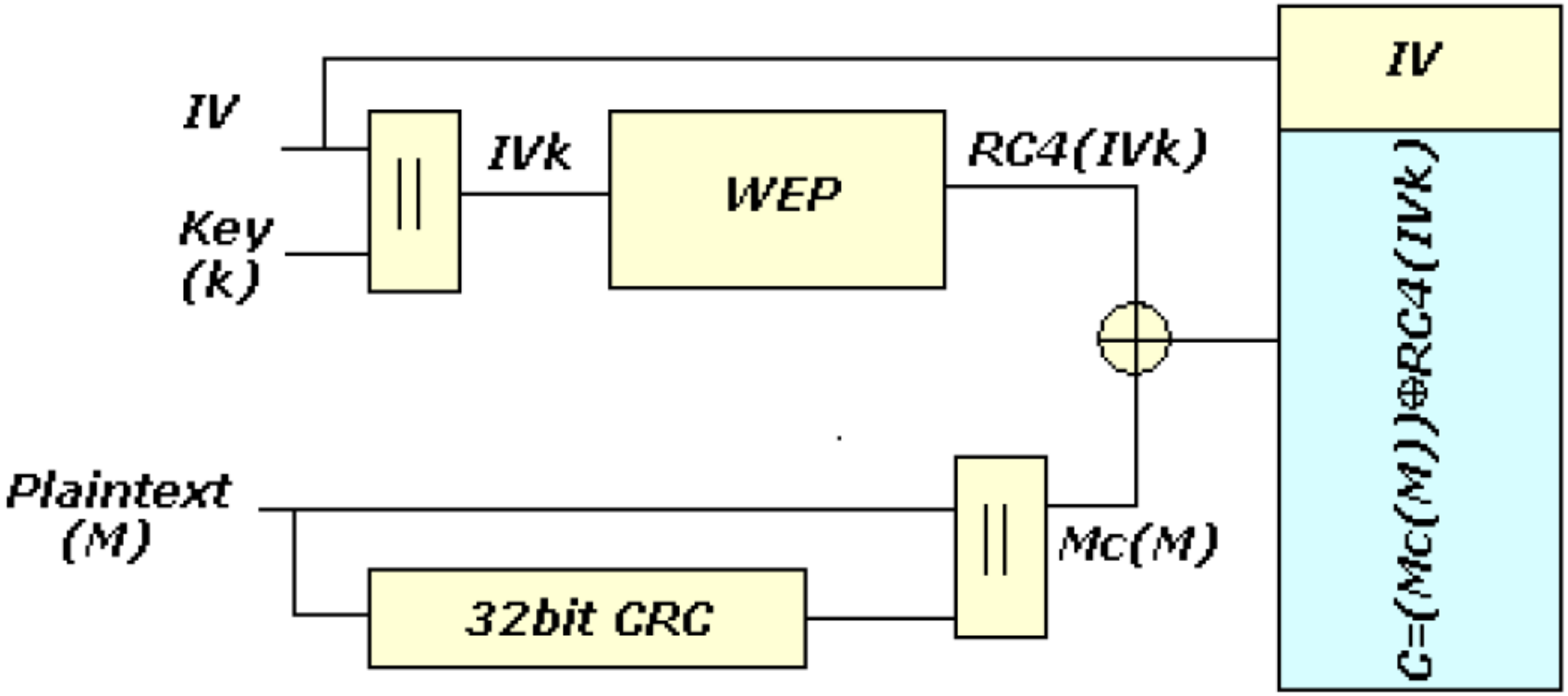
WEP: Come Funziona

- Per inviare un pacchetto di dati:
  - Dato il paylod M, viene calcolato il CRC di 32 bit c(M) che viene concatenato ad M → M·c(M)
  - La chiave k è concatenata all'IV determianto per il pacchetto → IV·K
  - L'algoritmo RC4 è inizializzato usando questo pacchetto e viene generata una sequenza di bytes → RC4(IV·k)
  - M·c(M) a questo punto e messo in xor con RC4(IV·k) → C = (M·c(M))⊕RC4(IV·k)
  - I 3 byte dell'IV sono trasmessi in chiaro (insieme con l'indice della chiave WEP)

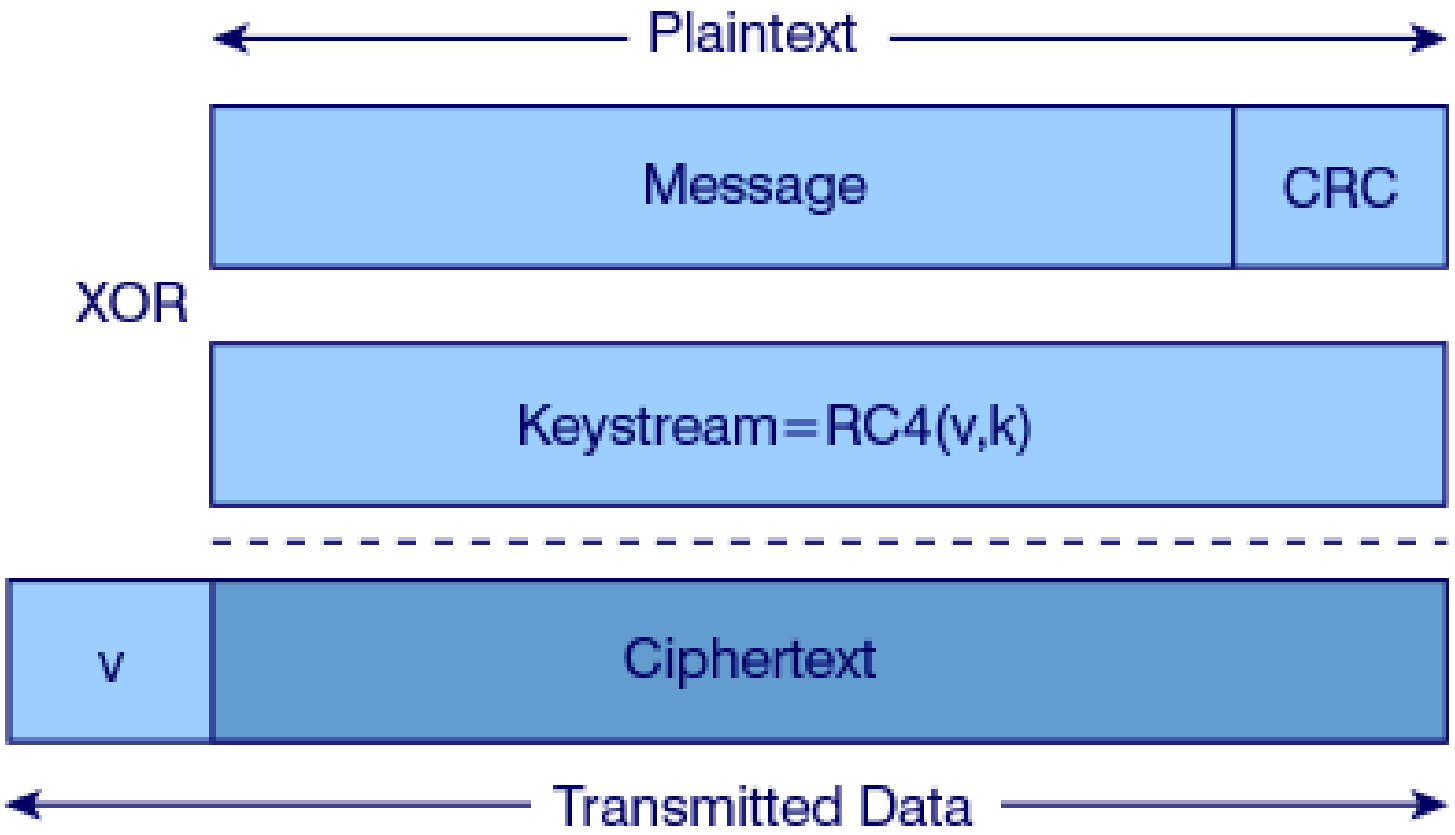
WEP: Come Funziona

- Il ricevente concatena l'IV ricevuto con la chiave WEP condivisa cosicché può ricostruire RC4(IV·k) → Questa è la ragione per cui l'IV deve essere trasmesso in chiaro
- Il ricevente decritta il payload e se il CRC coincide allora il pacchetto è valido altrimenti il pacchetto viene scartato

WEP: Come Funziona



WEP: Come Funziona



WEP: RC4

- Key Scheduling Algorithm
- RC4 utilizza un vettore di stato di 256 ottetti S[256] e due contatori i, j
- Inizializzazione dello stato:
  - S[n] = n, i=0, j=0
  - Il vettore temporaneo T di 256 ottetti si inserisce la chiave IV·K ripetendola se corta
  - Si scorre S scambiando gli elementi del vettore

```
for i = 0 to 255
    j = (j + S[i] + K[i mod 8]) mod 256
    scambia (S[i], S[j])
```

WEP: RC4

- Pseudo Random Generation Alghoritm. Generazione del keystream:
  - Per generare un ottetto z del keystream dallo stato corrente (S, i, j):
    i = (i + 1) mod 256
    j = (j + S[i]) mod 256
    scambia (S[i], S[j])
    t = (S[i] + S[j]) mod 256
    z = S[t]
  - Inizialmente i=0, j=0 e si scarta T
  - Il processo di generazione continuerà finche non ci sono più dati

Debolezze e vulnerabilità di WEP

WEP: Riutilizzo della codifica

- Se utilizziamo lo stesso IV, viene generata la stessa sequenza (keystream) di byte da RC4
- Crittando così due messaggi p1 e p2 abbiamo:
  - $C1 = P1 \oplus RC4(IV \cdot k)$
  - $C2 = P2 \oplus RC4(IV \cdot k)$
  - $C1 \oplus C2 = P1 \oplus RC4(IV \cdot k) \oplus P2 \oplus RC4(IV \cdot k)$
    $= P1 \oplus P2$
- Quindi con l'xor di due messaggi cifrati si ottiene l'xor dei due messaggi in chiaro

WEP: Riutilizzo della codifica

- Se si conosce uno dei due messaggi si ottiene l'altro
- Se si hanno molti messaggi codificati con lo stesso keystream è facile risalire ai messaggi originali
- I protocolli impongono molte similarità sui pacchetti!
- Non si devono riusare i keystream
- Notare che 24bit di IV vogliono dire 16777216 di diversi keystream: POCHI

WEP: Riutilizzo della codifica

- Lo standard raccomanda (ma non impone) che l'IV dovrebbe cambiare in modo casuale dopo ogni pacchetto trasmesso
- Alcune schede generano i 24 bit dell'IV utilizzando un counter azzerano l'IV ogni volta che sono inizializzate e poi incrementano il counter di 1 → aumentano la probabilità che la chiave sia riusata (i valori bassi di IV sono più probabili)

WEP: Attacchi di forza bruta

- Può utilizzare una lista di chiavi “facili”
- Analizzando l’intero spazio di ricerca dato dai 40 bit, ci possono volere circa 45 giorni → Non pratico per chiavi a 104 bit
- Bastano due pacchetti in generale (per essere sicuri che il CRC non coincida per caso anche con una chiave WEP sbagliata)

WEP: Attacchi basati su Weak IV

- S. Fluhrer, I. Mantin, A. Shamir hanno dimostrato che esistono delle debolezze nell'algoritmo di generazione delle chiavi in RC4 → "*Weakness in the Key Scheduling Algorithm of RC4"*
- L'attacco descritto nel loro articolo, oltre ad essere estremamente veloce, richiede un tempo che cresce linearmente con la lunghezza della chiave WEP!

WEP: Attacchi basati su Weak IV

- Il fatto che un larga parte della chiave (3 byte) sia trasmessa in chiaro aumenta la facilità di cracking:
  - Le prime tre iterazioni del KSA sono facilmente deducibili per il fatto che le prime tre cifre della chiave sono note (ricordate: l'IV è trasmesso in chiaro)!
- Si può vedere che c'è una probabilità del 5% che i valori in S[0] - S[3] non cambino dopo le prime 3 iterazioni del KSA

WEP: Attacchi basati su Weak IV

- È stato dimostrato che gli IV di un certo tipo sono soggetti ad essere crackati:
  (B+3:255:x)
  dove B è il byte della chiave segreta (la chiave WEP) che stiamo crackando
- Quindi per ogni byte della chiave ci sono 256 Weak IV

WEP: Attacchi basati su Weak IV

- I primi valori dei dati crittati è l'header SNAP (*Sub Network Attachment Point*). È uno standard (di livello 2) per la trasmissione di datagram IP su reti IEEE 802
- L'header non crittato è AA in esadecimale
- Xor dei primi dati crittati con AA ci da il primo byte del PRGA
- Questa informazione ci può consentire di ricostruire la prima cifra della chiave WEP se ho un Weak IV del tipo (3:255:x)

WEP: Attacchi basati su Weak IV

- Vediamo nel dettaglio il primo passo dell'algoritmo per generare il primo byte del keystream:

  i = (i + 1) mod 256 → i = 1

  j = (j + S[i]) mod 256 → j = S[1]

  scambia(S[i], S[j]) → scambia(S[1], S[S[1]])

  t = (S[i] + S[j]) mod 256 → t = S[1] + S[S[1]]

  z = S[t] → z = S[S[1] + S[S[1]]]

- Quindi il primo byte dipende da:

  S[1], S[S[1]] e S[S[1] + S[S[1]]]

WEP: Attacchi basati su Weak IV

- Vediamo i primi due passi della generazione del vettore S con IV (3:255:x):
- i = 0, j = 0

S →	0	1	2	3	4
T →	3	255	x	W_1	

- i = 0, j = (j + S[i] + T[i mod 8]) = (0+S[0]+T[0]) = 0+0+3 = 3 → swap(S[0],S[3])

S →	3	1	2	0	4
T →	3	255	x	W_1	

WEP: Attacchi basati su Weak IV

- i = 1, j = 3

S →	3	1	2	0	4
T →	3	255	x	W_1	

- i = 1, j = (j + S[i] + T[i mod 8]) = (3+S[1]+T[1]) = 3+1+255 = 3 → swap(S[1],S[3])

S →	3	0	2	1	4
T →	3	255	x	W_1	

WEP: Attacchi basati su Weak IV

- Al prossimo passo j = (3 + S[2] + T[2]) = (3 + (2 + x)) ovvero j avanza di x + 2 con x noto
- Ogni IV si comporta in maniera diversa dipendendo da x, ma siamo in grado di ricostruire la configurazione del vettore S
- Da qui in poi l'evoluzione di S dipende dalla chiave, e con una probabiltà del 5% (come dicevamo precedentemente) i primi 3 valori di S non cambiano

WEP: Attacchi basati su Weak IV

- Esistono anche altre famiglie di Weak IV
- Oltre il primo byte della chiave l'operazione si complica perché richiede di ciclare sul PRGA per più passi e quindi potremmo non essere più in grado di dedurre con una ragionevole probabilità le permutazioni di S

WEP: Attacchi basati su Weak IV

- Alcuni produttori di schede wireless hanno cominciato a produrre schede che evitano di utilizzare IV deboli
- Riduce ulteriormente lo spazio degli IV disponibili (qualche migliaio in meno)
- Notare che basta un solo client che non aggiri gli IV deboli ed è possibile portare a buon fine l'attacco

Airsnort: software per il crack delle chiavi WEP

Airsnort

- Esistono vari tools che consentono di determinare in modo automatico una chiave WEP
- Uno di questi è Airsnort, scaricabile all'indirizzo:
  http://airsnort.shmoo.com/
- È un programma linux ora anche windows
- Richiede che la scheda wireless sia in modalità monitor
- Funziona ad esempio con le schede Prism2, Orinoco e Cisco

Airsnort

- Una volta attivato, il programma cattura i pacchetti ed in contemporanea cerca di crackare la chiave WEP:
  - Tutti i pacchetti non data (eccetto i beacon) sono scartati
  - I pacchetti non crittati sono scartati
  - I pacchetti crittati sono selezionati e quelli ritenuti non interessanti sono scartati
- I pacchetti ritenuti interessanti sono i *Weak IV* individuati da Fluhrer, Mantin e Shamir (più *Weak IV* individuati successivamente)

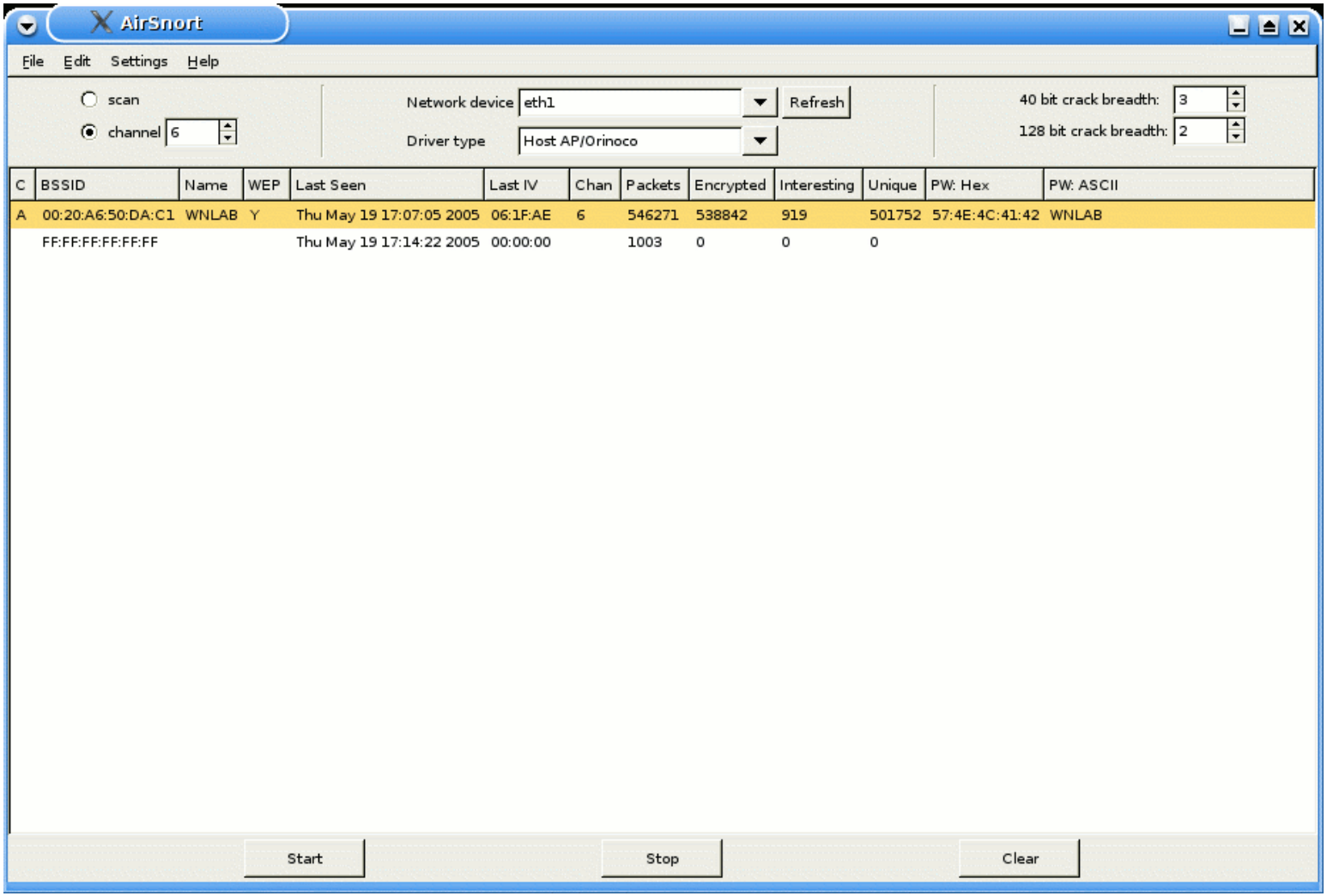
Airsnort

- Ogni 10 weak IV acquisiti, airsnort utilizza un attacco probabilistico
- Si può controllare quanto profondamente analizzare l'albero delle diverse possibilità
- Un valore *n* del parametro "breadth" indica che verranno provate gli *n* valori più probabili per ciascuna posizione della chiave
- Sono richiesti circa 1000 weak IV per una chiave a 64 bit e circa 2000 per una chiave a 128 bit

Airsnort

- Test di attacco effettuato utilizzando:
  - Access Point Avaya AP3
  - Due laptop per generare traffico
  - Un laptop con una scheda Netgear ed Airsnort
- Impostata una chiave WEP a 64 bit, ovvero 40 bit di chiave, ovvero 5 caratteri → WNLAB
- Dopo circa 15 minuti di acquisizione con circa 550.000 pacchetti acquisiti (di cui 540.000 criptati) e 919 Weak IV, la chiave è stata determinata!

Airsnort


AirSnort
File Edit Settings Help
scan
channel 6
Network device eth1
Refresh
Driver type Host AP/Orinoco
40 bit crack breadth: 3
128 bit crack breadth: 2
C BSSID Name WEP Last Seen Last IV Chan Packets Encrypted Interesting Unique PW: Hex PW: ASCII
A 00:20:A6:50:DA:C1 WNLAB Y Thu May 19 17:07:05 2005 06:1F:AE 6 546271 538842 919 501752 57:4E:4C:41:42 WNLAB
FF:FF:FF:FF:FF:FF Thu May 19 17:14:22 2005 00:00:00 1003 0 0 0
Start
Stop
Clear

Airsnort

- In tabella alcuni run di test:

Lunghezza Chiave	Pacchetti Acquisiti	Pacchetti Crittati	IV Deboli Individuati
40	283618	278860	120
40	546271	538842	919
40	283895	280098	100
40	283876	280057	102
104	285328	281596	104
104	285798	282076	850
104	575137	567385	933