Metro Olografix Crypto Meeting 2006
25 Marzo 2006
Pescara

“What's going wrong with WEP?”

Giulia Biagini <giulia@olografix.org>
Angelo Dell'Aera <buffer@antifork.org>

Di cosa parleremo...

- Stream ciphers
- RC4
- CRC-32
- WEP
- Problematiche del WEP
- Attacco di Fluhrer, Mantin e Shamir
- Attacco di KoreK (chopchop)
- WEP Hacking Live

Stream ciphers

- Uno *stream cipher* converte un plaintext in un ciphertext un byte alla volta
- Esso consta di un *keystream generator* la cui funzione è quella di generare uno stream di bit $k_1, k_2, \ldots k_i$
- Il keystream così generato viene XORato con uno stream di bit $p_1, p_2, \ldots p_i$ costituenti il plaintext

Stream ciphers

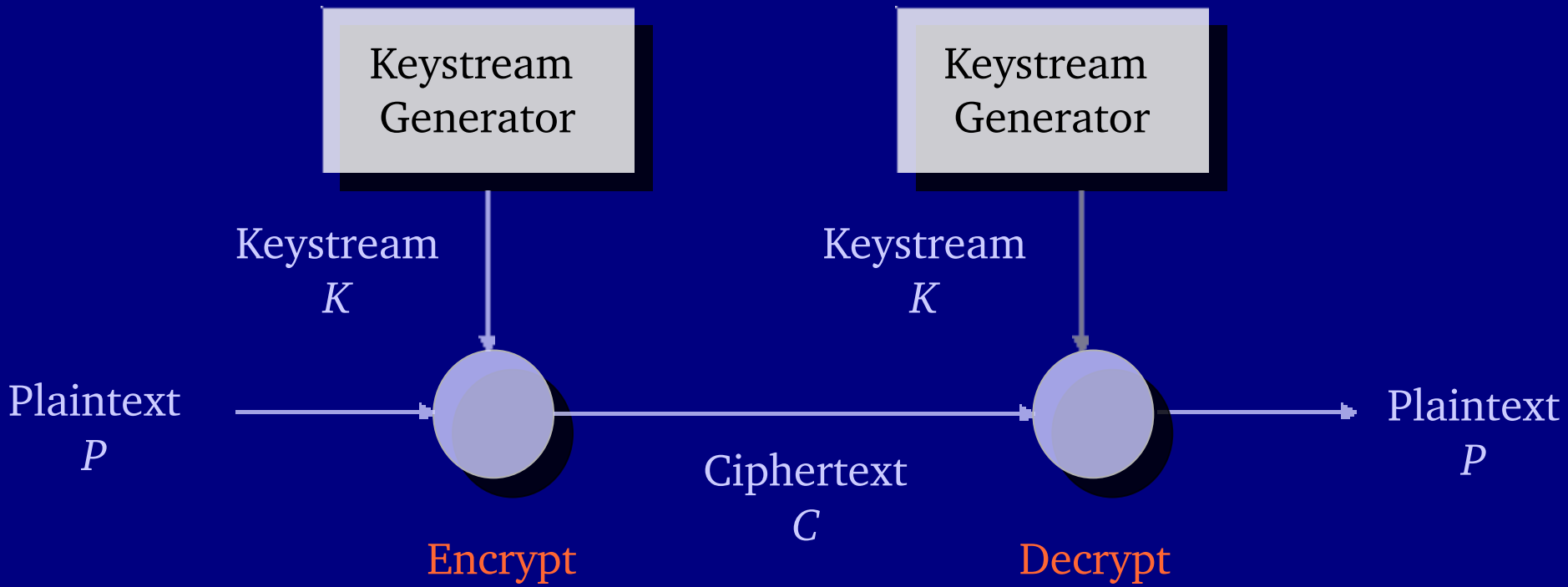
- Il risultato è che il bit i-esimo del ciphertext viene calcolato come

$$c_i = p_i \oplus k_i$$

- L'operazione di decrittazione è molto semplice a causa del fatto che

$$c_i \oplus k_i = p_i \oplus k_i \oplus k_i = p_i$$

Stream ciphers



Stream ciphers

- La sicurezza di uno *stream cipher* si basa interamente sulla qualità del *keystream generator*
- Se il keystream fosse una sequenza di zeri il ciphertext coinciderebbe con il plaintext
- Se il keystream fosse invece una sequenza realmente random avremmo per le mani un one-time pad e quindi la sicurezza perfetta
- Come al solito la verità è nel mezzo...

RC4

- RC4 consiste di due parti :

- *Key Scheduling Algorithm* (KSA) il cui scopo è quello di convertire una chiave iniziale (di dimensione tipicamente compresa tra 40 e 256 bit) in una permutazione iniziale *S*

- *Pseudo-Random Generation Algorithm* (PRGA) che utilizza la permutazione *S* per generare l'output stream

RC4
Key Scheduling Algorithm

KSA(K)

Initializiation:

```
For i = 0... N-1
    S[i] = i
j = 0
```

Scrambling:

```
For i = 0... N-1
    j = j + S[i] + K[i mod l]
    Swap(S[i], S[j])
```

RC4
Pseudo-Random Generation Algorithm

PRGA(K)

Initializiation:

$i = 0$

$j = 0$

Generation Loop:

$i = i + 1$

$j = j + S[i]$

Swap(S[i], S[j])

Output $z = S[S[i] + S[j]]$

RC4

- PRGA inizializza due indici *i* e *j* a 0 e successivamente esegue il *generation loop* durante il quale incrementa *i* di 1 e *j* in maniera pseudo-randomica (si noti come questa fase sia influenzata da KSA), scambia i valori di *S[i]* e *S[j]* e genera l'output *z*

- Si noti che ciascuna entry *S* viene swappata almeno una volta durante gli N round e, in questo modo, la permutazione *S* evolve durante la fase di PRGA

CRC-32

- CRC è l'acronimo di *Cyclic Redundancy Check*
- Scopo del CRC è quello di identificare l'integrità di un messaggio rilevando la presenza di errori all'interno dello stesso
- Il CRC-32 offre una protezione molto forte contro gli errori statisticamente più comuni in messaggi di lunghezza anche di qualche centinaia di byte

CRC-32

- Il CRC-32 basa il suo funzionamento su un polinomio di grado 32 indicato con $C(x)$
- Quando un sender vuole inviare un messaggio di $(N+1)$ bit esso in realtà invia $(N+1+k)$ con k grado del polinomio (nel caso del CRC-32 ovviamente $k=32$)

CRC-32

- Indicando il messaggio completo di CRC come $P(x)$, lo scopo del calcolo è fare in modo che $P(x)$ sia esattamente divisibile per $C(x)$
- Quando il receiver riceve $P(x)$ effettua la divisione per $C(x)$ e se non ci sono stati errori nella trasmissione il resto sarà nullo
- In caso contrario invece si è verificato un errore e in tal modo lo si riesce a rilevare

Il protocollo WEP

- Il protocollo WEP *(Wired Equivalent Privacy)* è parte integrante dello standard 802.11 definito dall'IEEE che definisce le specifiche per le wireless local area networks (WLAN)

- E' stato progettato come protocollo di sicurezza a data-link layer in grado di garantire lo stesso livello di sicurezza di una wired LAN

Obiettivi del protocollo WEP

Il protocollo WEP si pone essenzialmente due obiettivi:

- Proteggere la confidenzialità dei dati da link-layer eavesdropping
- Garantire un controllo sull'accesso alla rete

WEP

- I client wireless condividono una chiave WEP tipicamente a 40/104 bit con l'Access Point (AP)
- Alla chiave WEP vengono aggiunti 24 bit che costituiscono l'Initialization Vector (IV)
- E' un errore parlare di chiavi WEP a 64/128 bit in quanto l'IV viene trasmesso in chiaro!

WEP

- Ogni frame include un integrity check CRC-32 indicato con il termine *Integrity Check Value* (ICV)
- Se la verifica dell'ICV dovesse risultare invalida il frame verrebbe scartato in quanto corrotto
- Opzionalmente tutti i frame non crittati dovrebbero essere scartati

WEP

- Il protocollo WEP si basa sullo stream cipher RC4 per garantire la confidenzialità dei dati

- Lo stream cipher RC4 espande il seed <*IV, chiave*> in un keystream pseudorandom

- Le operazioni di crittazione e decrittazione sono identiche e consistono in una XOR con il keystream ottenuto dal seed

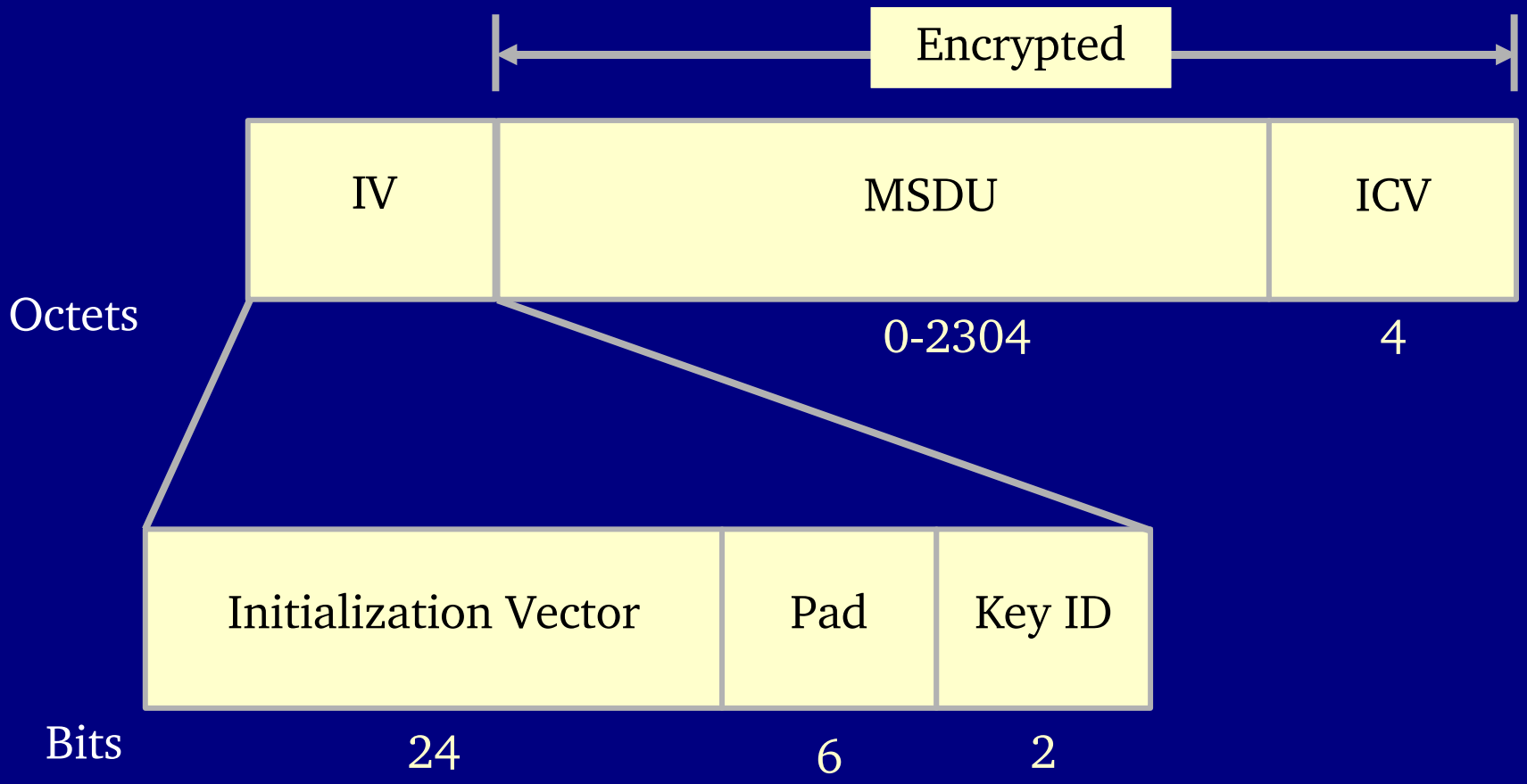
Costruzione di un frame WEP
Checksumming

- Si calcola il checksum *CRC(M)* del messaggio *M* da inviare
- Il messaggio *M* e il checksum *CRC(M)* vengono concatenati e vanno a formare il plaintext
  $P = <M, CRC(M)>$
- Si noti che *P* non dipende dalla chiave WEP *k*

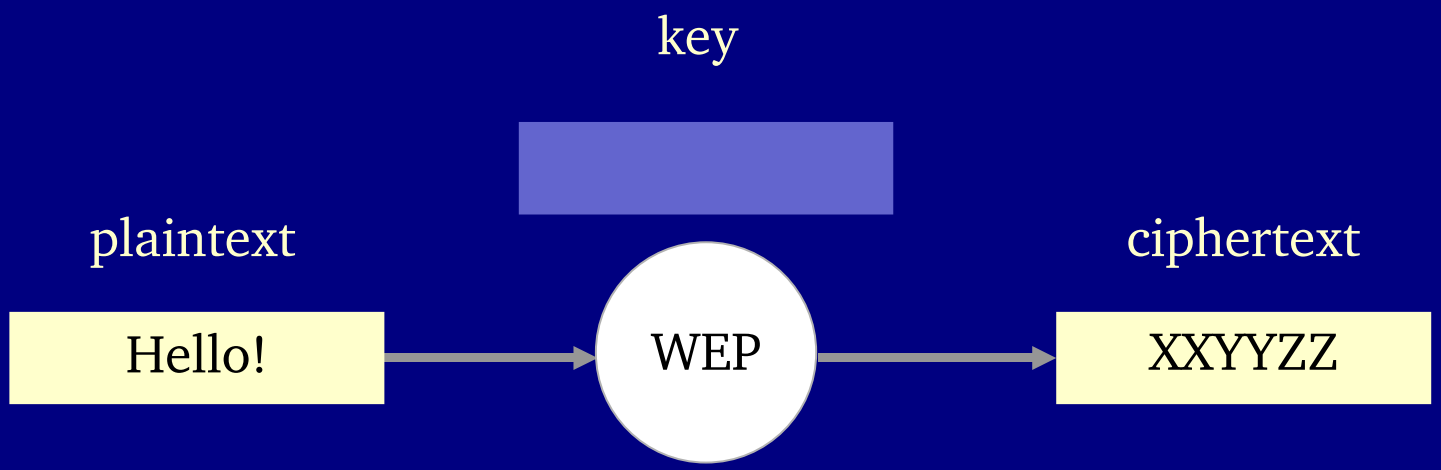
Costruzione di un frame WEP
Encryption

- Viene scelto un IV v che viene concatenato alla chiave k condivisa da client e AP
- Si genera un keystream mediante l'algoritmo RC4 che indicheremo con $RC4(v, k)$
- Viene generato il ciphertext come $C = P \oplus RC4(v, k)$

Struttura di un frame WEP



Initialization Vector



- Essendo il WEP basato su RC4, in assenza di IV, lo stesso plaintext produrrebbe sempre lo stesso ciphertext a causa del keystream sempre uguale

- Questo potrebbe consentire a un eavesdropper di osservare pattern nel ciphertext e ricavare il plaintext

Problema 1
Keystream Reuse

- Ipotizziamo che per crittare due plaintext P_1 e P_2 venga scelto lo stesso IV v. In tal caso avremmo

$$C_1 = P_1 \oplus RC4(v, k)$$
$$C_2 = P_2 \oplus RC4(v, k)$$

$$C_1 \oplus C_2 = (P_1 \oplus RC4(v, k)) \oplus (P_2 \oplus RC4(v, k)) =$$
$$P_1 \oplus P_2$$

Problema 1
Keystream Reuse

- In altre parole, mettendo in XOR due ciphertext crittati con lo stesso seed è possibile eliminare il keystream

- Successivamente è possibile ricavare uno dei due plaintext se si conosce l'altro mediante un attacco chosen-plaintext oppure ricavare entrambi con attacchi di tipo statistico

Problema 2
Checksum Linearity

- Il CRC-32 crittato è uno strumento valido per controllare la presenza di errori casuali ma non di quelli introdotti di proposito!

- Infatti il CRC-32 è lineare il che implica che

$$CRC(X \oplus Y) = CRC(X) \oplus CRC(Y)$$

Problema 2
Checksum Linearity

- E' possibile trovare un nuovo ciphertext C' che venga decrittato in M' con $M' = M \oplus \mu$ con μ scelto arbitrariamente dall'attaccante

$$C' = C \oplus <\mu, CRC(\mu)>$$

$$= RC4(v,k) \oplus <M, CRC(M)> \oplus <\mu, CRC(\mu)>$$

$$= RC4(v,k) \oplus <M \oplus \mu, CRC(M) \oplus CRC(\mu)>$$

$$= RC4(v,k) \oplus <M', CRC(M \oplus \mu)>$$

$$= RC4(v,k) \oplus <M', CRC(M')>$$

Problema 3
Traffic Injection

- Se un attaccante riuscisse ad avere a disposizione sia il plaintext sia il ciphertext conoscerebbe il keystream e potrebbe usarlo per iniettare pacchetti usando lo stesso IV

- E' possibile montare un attacco di questo tipo mandando del traffico in chiaro noto verso la rete wireless e sniffando alla ricerca del ciphertext

Problema 4
Traffic Reinjection

- WEP non implementa alcuna forma di protezione nei confronti dei *replay attacks*
- E' quindi possibile reiniettare più volte un frame crittato e questo sarà considerato valido
- Questo problema viene sfruttato nell'attacco attivo più efficace al WEP ossia l'*ARP request reinjection*

Attacco di Fluhrer, Mantin e Shamir (FMS)

- L'attacco FMS è un attacco passivo proposto nel paper "*Weakness in the Key Scheduling Algorithm of RC4*" di Fluhrer, Mantin e Shamir

- L'attacco sfrutta i punti deboli presenti nell'algoritmo di pianificazione delle chiavi di RC4 e l'impiego degli IV

Attacco di Fluhrer, Mantin e Shamir (FMS)

- L'attacco FMS poggia le sue basi su un'analisi crittanalitica di RC4
- Nel paper Fluher, Mantin e Shamir formalizzano quella che gli autori definiscono *“KSA Invariance Weakness”*

Attacco di Fluhrer, Mantin e Shamir (FMS) KSA Invariance Weakness

- A seguito di un'analisi crittanalitica di RC4 è possibile infatti dimostrare che esiste una classe particolari di chiavi che hanno la proprietà di far sì che KSA trasformi pattern nella chiave *K* in pattern corrispondenti nella permutazione iniziale *S*

- Questi *"weak key patterns"* possono, in determinate condizioni, propagarsi fino all'output di RC4

Attacco di Fluhrer, Mantin e Shamir (FMS)

- I risultati della crittanalisi di RC4 applicati a WEP mostrano che valori “deboli” di IV *(weak IVs)* lasciano trapelare informazioni sulla chiave

- Raccogliendo un numero sufficiente di pacchetti caratterizzati da un *weak IV* è possibile individuare la chiave nel caso in cui il primo byte del keystream sia noto

Attacco di Fluhrer, Mantin e Shamir (FMS)

- Fortunatamente il primo byte di un frame 802.11 è noto essendo il primo byte dell'header 802.2 LLC (DSAP) il cui valore è 0xAA
  - Il primo byte del keystream quindi può essere ottenuto applicando l'operatore XOR al primo byte del frame crittato con 0xAA

Attacco di Fluhrer, Mantin e Shamir (FMS)

- Si supponga che l'IV preceda la chiave nella costituzione del seed RC4 e si supponga di conoscere le prime *A* parole della chiave K[3], ..., K[A + 3] (con A inizialmente uguale a 0 ovviamente)
- Si noti che questo scenario rispecchia perfettamente l'algoritmo usato dal WEP visto in precedenza

Attacco di Fluhrer, Mantin e Shamir (FMS)

- Si supponga inoltre di osservare una serie di IV aventi la seguente struttura

$$(A + 3, N - 1, X)$$

(con $N = 256$) per un numero di valori di X approssimativamente pari a 60

- Si può dimostrare che in queste condizioni è possibile ricavare K[A + 3] con una probabilità di successo maggiore di 0.5

Attacco di KoreK

- Un talentuoso e giovane hacker di nome KoreK ha teorizzato un attacco attivo particolarmente efficace nei confronti del WEP
- Questo attacco si basa su una vulnerabilità dell'ICV e l'idea alla base di questo attacco è molto semplice ma allo stesso tempo tremendamente efficace

Attacco di KoreK

- Abbiamo visto che il messaggio che viene crittato si ottiene come la concatenazione del messaggio vero e proprio e di un CRC calcolato sul messaggio stesso

$$P = <M, CRC(M)>$$

e questo plaintext viene poi usato per generare il ciphertext mediante una XOR con lo stream RC4

$$C = P \oplus RC4(v, k)$$

Attacco di KoreK

- Se al ciphertext così ottenuto viene troncato l'ultimo byte otterremo ovviamente un messaggio invalido

- E' però possibile dimostrare matematicamente che mettendo in XOR questo messaggio invalido con un determinato valore è possibile ottenere un messaggio valido e la cosa interessante è che questo valore dipende soltanto dal byte troncato del cleartext

Attacco di KoreK

- L'attacco chopchop di KoreK tronca l'ultimo byte del frame, assume che il suo valore sia 0, corregge il frame di conseguenza e lo reinietta nell'AP
- Se il frame iniettato è valido esso verrà “broadcastato” dall'AP altrimenti verrà droppato

Attacco di KoreK

- Se il frame viene droppato l'attacco procede provando tutti i possibili valori per ricavare il byte del cleartext (256 tentativi al massimo) fino a ottenere un frame valido
- L'attacco procede alla stessa maniera per tutti gli altri byte del frame
- Il risultato è che è possibile decrittare il frame in analisi senza conoscere la chiave!

Aircrack

- Aircrack è lo strumento principe per realizzare attacchi attivi e passivi al WEP
- Esso implementa gli attacchi statistici sviluppati da KoreK nonchè una versione particolarmente ottimizzata dell'attacco FMS

Live Session!

Riferimenti

- Fluhrer, Mantin, Shamir
  "Weakness in the Key Scheduling Algorithm of RC4"
  (Eight Annual Workshop on Selected Areas in Cryptography, 2001)

- Borisov, Goldberg, Wagner
  "Intercepting Mobile Communications : The Insecurity of 802.11"
  (Proceedings of the Seventh Annual International Conference on Mobile Computing And Networking, 2001)

- KoreK
  "Quick Explanation of the Inner Workings of Chopchop"
  http://www.netstumbler.org/archive/index.php/t-12277.html

Ringraziamenti

- Guido “Zen” Bolognesi per il prezioso materiale

- Scott Fluhrer, Itsik Mantin, Adi Shamir, Christophe Devine e Korek per aver reso il discorso sicurezza in ambito 802.11 così piccante...