Keamanan Nirkabel: Kelemahan WEP

Ibrohim Kholilul Islam
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jl. Ganesha 10 Bandung 40132, Indonesia
13513090@std.stei.itb.ac.id

Abstract—

Keywords— WEP, Kelemahan WEP, RC4

I. PENDAHULUAN

Perihal keamanan, komunikasi kabel dan komunikasi nirkabel cukup berbeda. Perbedaan ini muncul dari perbedaan layer fisik. Kabel hanya menghubungkan dua buah perangkat, sedangkan Wireless menggunakan satu media secara bersama-sama. Akibatnya seluruh pesan yang dikirimkan melalui Wireless, dapat di-'dengar' oleh seluruh perangkat yang terhubung dengan jaringan tersebut.

IEEE 802.11 merupakan standar untuk Wireless LAN, yang banyak ditemui di tingkat konsumen. WEP merupakan salah satu protokol untuk melakukan pengamanan terhadap lalu lintas data. Namun, pada 2001 oleh Fluhrer, Mantin dan Shamir ditemukan kelemahannya dan dianggap tidak aman lagi. Pada 2003, standar WEP digantikan dengan WPA oleh IEEE.

A. Notasi

Bagian ini dibuat untuk memberikan kemudahan dalam membaca makalah ini, disini akan dibahas terlebih dahulu notasi yang digunakan.

Untuk melakukan *assignment* pada sebuah peubah (variabel) digunakan `x ← c` yang berarti x diisi dengan nilai c. Oada perulangan dengan peubah, digunakan `i traversal [0..255]` diikuti dengan perintah di baris selanjutnya yang menjorok ke dalam. Array dituis dengan menggunakan tanda [.] seperti pada bahasa-bahasa pemrograman umumnya. Operator biwise XOR dilambangkan dengan menggunakan A ⊕ B yang berarti A di-XOR-kan dengan B.

B. RC4

Dalam WEP, enkripsi sebagian besar berdasarkan algoritma streamkey generator RC4. RC4 memiliki dua mekanisme: RC4-KSA, yang mengubah 1 – 256 bytes untuk melakukan permutasi pada S yang bernilai 0 – 255.

Algoritma 1: RC4-KSA

```
1 | i traversal [0..255]:
2 |   S[i] ← i
3 | j ← 0
4 | j traversal [0..255]:
5 |   j ← j+S[i]+K[i mod len(K)] mod 256
6 |   swap(S, i, j)
7 | i ← 0
8 | j ← 0
```

Algoritma 2: RC4-PRGA

```
1 | i ← i + 1 mod 256
2 | j ← j + S[i] mod 267
3 | swap(S, i, j)
4 | ← S[S[i] + S[j] mod 256]
```

RC4 menggunakan *state* i dan j sebagai penunjuk pada larik S. Setiap kali RC4-PRGA dijalankan algoritma ini mengembalikan sebuah byte dan mengubah state i, j, dan S.

C. CRC32

CRC adalah fungsi hash satu arah yang digunakan oleh protokol WEP untuk memastikan integritas data. Prinsip yang digunakan CRC adalah menggunakan prinsip pembagian:

1. Untuk menghitung n-bit CRC, dibutuhkan pola n+1 bit sebagai konstanta pembagi CRC (*c*).
2. Pesan diberikan padding sepanjang n-bit
3. Untuk setiap bit mulai dari MSB (*most significant bit*):
4. Jika bit tersebut bernilai 1, lakukan XOR dengan konstantan *c*. Jika tidak, lewati
5. Ulangi langkah 4 hingga seluruh bit pesan menjadi 0.
6. Sisa nilai pada padding merupakan nilai kembalian dari CRC.

Satu catatan penting mengenai CRC adalah nilai CRC bersifat linear terhadap pesan asli, dan tidak memerlukan kunci untuk melakukannya.

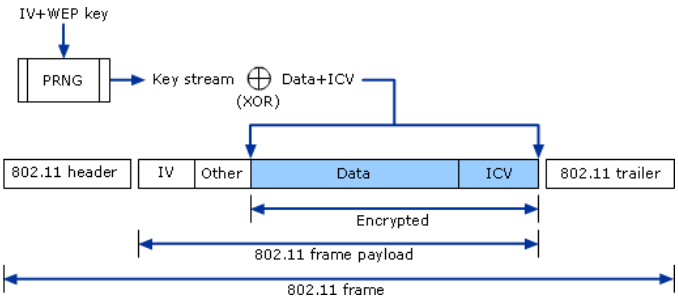
Contoh 1 Ilustrasi CRC3

```
Misalkan:
11010011101100

11010011101100 000 <--- diberikan padding 3 bit
1011               <--- konstanta pembagi (4 bits) = x³ + x + 1
01100011101100 000 <--- hasil
 1011              <--- konstanta pembagi ...
00111011101100 000
  1011
00010111101100 000
   1011
00000001101100 000 <--- catatan: bit 0 dilewati
       1011
00000000110100 000
        1011
00000000011000 000
         1011
00000000001110 000
          1011
00000000000101 000
           101 1
-----------------
00000000000000 100 <--- hasil
```

D. WEP

WEP (Wired Equivalent Privacy) pertama dipublikasikan sebagai bentuk ekamanan data pada standar IEEE 802.11 pada tahun 1999. Cara kerja WEP secara umum adalah dengan melakukan enkripsi pada pesan.



https://i-technet.sec.s-msft.com/dynimg/IC196394.gif

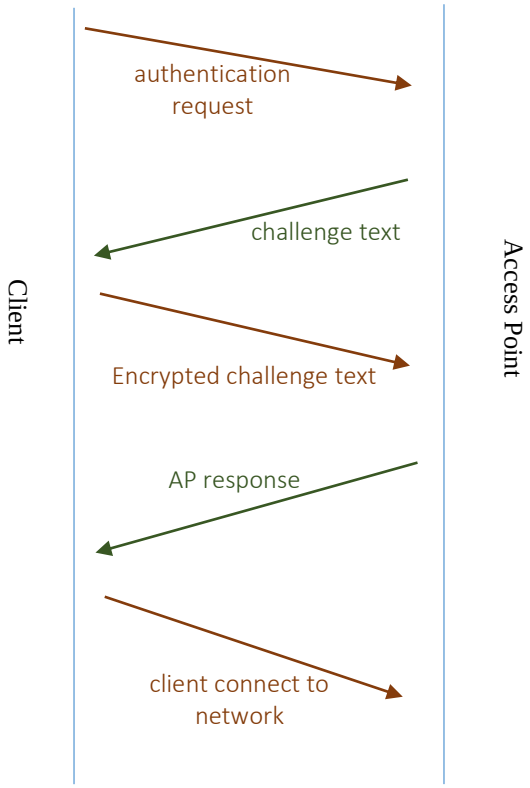
Enkapsulasi dilakukan dengan menambahkan IV, keyId dan ICV:

1. IV (Initialization Vector) sebesar 24-bit, IV adalah blok dari bit-bit yang digunakan untuk memulai RC4 untuk menghasilkan keystream yang unik untuk tiap sesi
2. Padding dengan format big endian sebesar 6 bit
3. 2 bit ide kunci.
4. Data merupakan hasil enkripsi dengan menggunakan melakukan XOR pada plainteks dengan keystream RC4.
5. Integrity Check Value (ICV) sebesar 32 bit, ICV pada WEP menggunakan CRC32.

Terdapat 2 cara melakukan autentikasi pada WEP:

- Open System Authentication
- Shared Key Authentication

Pada makalah kali ini, akan difokuskan membahas *Shared key authentication*. *Shared key authentication* dilakukan dengan proses jabat tangan 4 arah dengan challenge. Langkah-langkah tersebut adalah:



1. Client yang akan melakukan koneksi meminta otentikasi dari Access Point.
2. Access Point mengirimkan challenge dengan berupa plainteks
3. Client harus melakukan enkripsi terhadap challenge menggunakan kunci WEP yang digunakan oleh Access Point dan mengirimkannya kembali kepada Access Point.
4. Access Point melakukan melakukan dekripsi terhadap pesan yang dikirimkan dan melakukan perbandingan dengan plainteks challenge yang diberikan. Apabila hasilnya sama, maka Client AP membalas client untuk memulai koneksi.

II. Kesalahan pada WEP

A. *IV terlalu pendek dan berupa plaintext*

IV hanya berupa 24-bit yang dikirimkan bersama pesan. String 24-bit ini digunakan untuk membangkitkan keystream dengan menggunakan RC4 dan merupakan ukuran yang terlalu kecil untuk keperluan kriptografi.

B. *IV terulang pada waktu yang cukup singkat*

Penggunaan IV yang sama, akan menghasilkan keystream yang sama. Karena IV cukup pendek dan hanya memiliki 2^{24} = 16777216 kemungkinan, maka IV dalam selang waktu yang singkat akan digunakan kembali.

C. *IV digunakan sebagai key RC4*

Karena penyerang mengetahui 24-bit dari setiap paket, dan dikombinasikan dengan kelemahan RC4, mengakibatkan penyerang mudah melakukan analisis.

D. *WEP tidak memiliki pengecekan integritas kriptografi*

Penggunaan CRC sebagai pengecekan integritas paket, yang dibarengi dengan penggunaan stream cipher merupakan kombinasi yang berbahaya dari sisi kriptografi.

III. Serangan yang Telah Dipublikasi

A. *Serangan FMS*

Fluhrer, Mantin, dan Shamir mempublikasikan serangan untuk memperoleh kembali kunci dari WEP. Serangan tersebut berdasarkan:

Penyerang dapat mendengar secara pasif lalu lintas data pada jaringan yang dilindungi WEP untuk mendapatkan cukup banyak paket beserta Initialization Vector (IV) yang digunakan pada paket tersebut. Karena beberapa karakter awal dari isi pesan sebuah paket biasanya mudah ditebak, penyerang dapat memperoleh kembali kunci.

Serangan ini dilakukan dengan mencari IV lemah sebanyak-banyaknya sebagai alat bantu dalam memecahkan kunci yang digunakan. Dalam 16 juta kemungkinan IV, terdapat sekitar 9000 kunci lemah, dengan metode FMS, dibutuhkan antara 1500 sampai 5000 paket data dengan IV lemah yang harus dicapture.

Salah satu versi yang mudah untuk menjelaskan serangan ini adalah dengan mengasumsikan kunci sesi berupa IV | kunci. Dengan hanya memperhatikan salah satu IV dengan bentuk (3, 255, X), pada awal algoritma RC4-KSA, j akan bernilai 0 dan S bernilai:

0, 1, 2, 3, 4, 5, ... 255.

Pada iterasi pertama baris ke 4 pada Algoritma 1, dengan K[0] = 3, nilai j = (0 + 0 + 3) mod 256 yakni 3 dan kondisi S sekarang adalah

3, 1, 2, 0, 4, 5, ... 255.

Pada iterasi ke 2, dengan nikai K[1] = 255, maka j = (3 + 1 + 255) mod 256 yakni 3 dan kondisi S sekarang adalah

3, 0, 2, 1, 4, 5, ... 255.

Jika nilai X diketahui maka kita tahu nilai S dan j sebagai fungsi dari X.

Dengan mengasumsikan byte S[0], S[1], dan S[3] tidak berubah, yani :

P(1 byte tidak berubah pada pertukaran acak) = (1 – 1/N)
Dengan N adalah panjang kunci keystream

P(1 byte tidak berubah pada N pertukaran acak) = $(1 - 1/N)^N$
P(3 byte tidak berubah pada N pertukaran acak) = $((1 - 1/N)^N)^3$

$$\exp(x) = \lim_{n\to\infty} \left(1 + \frac{x}{n}\right)^n$$

P(3 byte tidak berubah pada Npertukaran acak) = $(e^{-1})^3 = e^{-3}$

Dengan menggunakan byte-byte tersebut, memudahkan menebak K[4] dan seterusnya.

B. Serangan KoreK Chopchop (Chopping attack)

Serangan yang mampu memberikan hasil berupa plaintext dari ciphertext yang didapatkan. Serangan ini diusulkan oleh seorang member dari forum netstumbler.org, dengan username KoreK yang dipublikasikan pada tanggal 14 September 2004.

Prosedur serangan chopchop ini adalah dengan melakukan dekripsi dari paket byte per byte, kemudian byte terakhir dipotong dan diasumsikan bernilai 0, lalu memperbaiki paket dan mengirimkannya kembali ke access point. Perbaikan paket dilakukan dengan cara menghitung suatu nilai tertentu berdasarkan perkiraan byte, kemudian dilakukan operasi XOR dengan byte tersebut, dan menjadikannya paket baru untuk dikirimkan. Apabila asumsi dengan nilai 0 benar, maka paket akan diangggap benar oleh access point, kemudian akan dilakukan broadcast terhadap paket. Jika asumsi salah, maka paket akan ditolak oleh access point. Bila hal itu terjadi, maka Chopchop akan mencoba secara incremental dengan semua kemungkinan yang ada.

Chopchop akan mencoba 256 kemungkinan, (1-255), oleh karena itu, perlu suatu cara untuk mengenali paket yang ditransmisikan kembali oleh access point. Chopchop menyandikan perkiraannya pada byte terakhir pada paket, sehingga tidak memerlukan waktu untuk menunggu jawaban yang tidak memungkinkan.

Ketika access point mulai mengirimkan paket dengan dst-mac yang sesuai dengan pencarian, byte yang diuji akan bergeser, dengan byte terakhir dianggap konstan. Cara ini membutuhkan pengumpulan paket yang masif sehingga diperlukan cara untuk mempercepat pengumpulan paket dengan IV yang unik.

C. IV Collisions

Prosedur IV Collisions dilakukan dengan cara mengintersepsi semua paket yang melewati jaringan sampai mendapatkan dua paket dengan IV yang sama. Kemudian pada paket tersebut dilakukan operasi XOR, yang menghasilkan packet sniffer. Packet sniffer berisikan hasil XOR dari dua plaintext dengan cara sebagai berikut.

$$RC4(k,x) \oplus RC4(k,y) = x \oplus y$$

dengan :

k = key

x = plaintext 1

y = plaintext 2

Dengan melakukan analisis dari beberapa paket dengan IV yang sama, penyerang dapat menghasilkan plaintext yang diinginkan dan membaca isinya. Hasil plaintext dari IV collisions dapat digunakan untuk menciptakan “ciphertext”, yang dapat digunakan pada serangan Man In the Middle. Hal ini dilakukan dengan cara:

$$RC4(k,x) \oplus x \oplus y = RC4(k,y)$$

Serangan ini sangat mudah dilaksanakan pada jaringan nirkabel dengan IV statis. Hal ini dikarenakan prosedur ini hanya membutuhkan beberapa paket untuk mendapatkan nilai IV tersebut. Dari nilai IV tersebut dapat dibangkitkan nilai k, sehingga didapatkan plaintext yang digunakan. Informasi yang didapatkan dari plaintext tersebut juga dapat digunakan untuk melakukan serangan dan mengendalikan traffic data yang melewati jaringan.

D. Dictionary Building Attack

Prosedur Dictionary building attack dibangun berdasarkan hasil dari IV Collisions. Setelah diketahui beberapa plaintext dan ditemukan keystreamnya, dengan menggunakan IV yang diketahui, penyerang membangun tabel yang berisi IV dan keystream yang bersesuaian. Kemudian pada tabel tersebut dilakukan otomatisasi dekripsi dari IV dan keystream yang bersesuaian. Hal ini mempermudah penyerang dalam melakukan dekripsi paket yang diinginkan, kemudian IV dan keystream yang ditemukan akan ditambahkan ke dalam tabel.

Metode serangan ini menjadi semakin kuat dan mudah digunakan dengan bertambahnya data IV dan keystream, namun dibutuhkan kapasitas harddisk yang tidak sedikit. Pada umumnya sebuah tabel yang cukup lengkap dapat menggunakan space sebesar 15GB untuk penyimpanannya.

E. Replay Attack

Replay attack adalah metode serangan dimana data transmisi yang sudah dianggap valid disimpan oleh penyerang dan kemudian digunakan lagi oleh penyerang dengan berpura-pura menjadi client dari sebuah access point. Serangan ini tidak digunakan untuk memenuhi traffic dan mengumpulkan paket, namun untuk mendapatkan akses ke dalam jaringan.

F. Traffic Injection

Traffic Injection adalah metode untuk mempersingkat waktu pengumpulan paket dengan IV yang diinginkan. Metode ini dilakukan dengan mengumpulkan paket ARP, kemudian paket tersebut dikirimkan kembali ke access point. Metode ini membutuhkan spesifikasi alat dan software tertentu, bahkan untuk beberapa alat, dibutuhkan driver versi tertentu.

IV. ANALISIS IMPLEMENTASI PADA AIRCRACK-NG

Pada tahun 2005, Andreas Klein mempresentasikan sebuah analisis terhadap RC4 stream cipher yang menunjukan lebih banyak korelasi antara keystream RC4 dan kunci tersebut. Erik Tews, Ralf-Philip Weinmann, dan Andrei Pychkine

menggunakan analisis ini untuk membuat aircrack-ptw, yaitu sebuah perangkat untuk memecahkan 104-bit RC4 yang digunakan di 128-bit WEP dalam waktu 1 menit.

A. *Korelasi Klein pada RC4*

1) Teorema Klein

Misalkan S adalah permutasi acak dari bilangan [0 .. n-1]

$$Pr(S[S[i] + x] + x = i) = 2/n$$

$$Pr(S[S[i] + x] + x = c) = \frac{n-2}{n} * (n-1), \qquad c \neq \text{i}$$

Misalkan telah diketahui suatu paket yang 15 byte pertamanya telah diketahui plainteksnya. Paket tersebut akan dioperasikan sebagai berikut

$$X[i] = ciphertext[i] \oplus plaintext[i], i \in \{0,1, \dots, 14\}$$

Karena telah diketahui nilai IV = (K[0], K[1], K[2]), maka dapat dilakukan algoritma RC4 sebanyak 3 kali, dan akan didapatkan S_3 dan j_3. Dari S_3 dapat dengan mudah dilakukan inversi seperti cara pada subbab sebelumnya.
Untuk i = 3, rumusan ... akan menjadi

Kode Sumber 1 aircrack-ptw-lib.c (line 611 - 638)

```
1  for (i = 0; i < keylen; i++) {
2    memset(&table[i][0], 0, sizeof(PTW_tableentry) * n);
3    for (j = 0; j < n; j++) {
4      table[i][j].b = j;
5    }
6    for (j = 0; j < state->packets_collected; j++) {
7      // fullkeybuf[0] = state->allsessions[j].iv[0];
8      memcpy(fullkeybuf, state->allsessions[j].iv, 3 *
9  sizeof(uint8_t));
10     guesskeybytes(i+3, fullkeybuf, state->allsessions[j].keystream,
11 guessbuf, 1);
12     table[i][guessbuf[0]].votes += state->allsessions[j].weight;
13   }
14   qsort(&table[i][0], n, sizeof(PTW_tableentry), &compare);
15   j = 0;
16   while(!validchars[i][table[i][j].b]) {
17     j++;
18   }
19   // printf("guessing i = %d, b = %d\n", i, table[0][0].b);
20   fullkeybuf[i+3] = table[i][j].b;
21 }
```

2) Inversi Permutasi

Dari Algoritma 1 baris ke 5 dan 6, didapatkan persamaan
$j_{i+1} = j_i + S_i[i] + K[i \bmod l]$
$S_{i+1}[i] = S_i[j_i+1]$
Dengan mensubtitusi , didapatkan persamaan
$S_{i+1}[i] = S_i[j_i + S_i[i] + K[i \bmod l]]$

Dengan
$h = S_i+1[i]$
$g = j_i + S_i[i] + K[i \bmod l]$

maka $S[g] = h$ jika dan hanya jika $S^{-1}[h]=g$

3) Serangan Klein

Serangan Klein berdasarkan pada teoremanya, dengan menggunakan IV yang telah diketahui,

$$\Pr(K[3] = S_3^{-1}\left[3 - X[2]\right] - (S_3[3] + j_3)) \approx 1.36/n$$

Dimana $k_0 = S_3^{-1}\left[3 - X[2]\right] - (S_3[3] + j_3)$
Dengan menghitung nilai k_0 dan menyimpannya sebagai kandidat dari Rk[0] dengan probabilitas 1.36/n, namun dengan probabilitas 1-1.36/n, Rk[0] dapat memiliki nilai yang berbeda dari k_0 . Kemudian bukti-bukti lain tentang Rk[0] dapat ditentukan dengan paket yang dikirim di antara AP yang sama namun memiliki IV yang berbeda, yang memiliki informasi yang berbeda pula. Ketika voting yang didapat sudah mencukupi, maka dapat dipilih nilai k_0 yang memiliki probabilitas yang paling tinggi. Klein memperkirakan bahwa 25000 IV yang unik dapat mendapatkan kembali informasi byte Rk[0].
Setelah memilih k_0 yang memiliki probabilitas tertinggi, maka dapat diasumsikan $K[3] = k_0$, yang memungkinkan dilakukan algoritma RC4 untuk setiap IV. Dengan menggunakan kumpulan IV yang sama, dapat dicari byte Rk[1]. Dengan menggunakan pendekatan ini, maka dapat ditentukan semua

byte dari Rk, sehingga dapat dicoba untuk melakukan dekripsi cipherteks yang diketahui sebagian plainteksnya.
Pada kasus dimana IV unik terlalu sedikit, maka kandidat untuk Rk[i] mungkin bukanlah yang paling tinggi probabilitasnya, sehingga dilakukan perhitungan ulang dengan byte lain Rk[i+1], ..., Rk[l-1] untuk setiap Rk[i] yang baru. Iterasi ini akan terus berulang sampai mendapatkan Rk yang benar.

B. Implementasi Klein pada Aircrack-ng

Algoritma Klein diimplementasi pada Aircrack-NG pada file: aircrack-ptw-lib.c (Kode sumber 1)

```
/*
 * Guess which key bytes could be strong and start actual computation of the key
 */
```

Kode Sumber 2 aircrack-ptw-lib.c (line 349-376)

```
1  static void guesskeybytes(int ivlen, uint8_t * iv, uint8_t *
   keystream, uint8_t * result, int kb) {
2        uint32_t state[n];
3        uint8_t j = 0;
4        uint8_t tmp;
5        int i;
6        int jj = ivlen;
7        uint8_t ii;
8        uint8_t s = 0;
9        memcpy(state, &rc4initial, sizeof(rc4initial));
10       for (i = 0; i < ivlen; i++) {
11             j += state[i] + iv[i];
12             tmp = state[i];
13             state[i] = state[j];
14             state[j] = tmp;
15       }
16       for (i = 0; i < kb; i++) {
17             tmp = jj - keystream[jj-1];
18             ii = 0;
19             while(tmp != state[ii]) {
20                   ii++;
21             }
22             s += state[jj];
23             ii -= (j+s);
24             result[i] = ii;
25             jj++;
26       }
27       return;
28 }
```

```
int PTW_computeKey(PTW_attackstate * state, uint8_t * keybuf, int keylen, int testlimit, int * bf, int validchars[][n], int attacks);
```

pada bagian:

```
for (j = 0; j < state->packets_collected; j++)
```

dilakukan iterasi untuk seluruh paket yang untuk kemudian dilakukan penebakkan:

```
guesskeybytes(i+3, fullkeybuf, state->allsessions[j].keystream, guessbuf, 1);
```

kemudian dilakukan vote:

```
table[i][guessbuf[0]].votes += state->allsessions[j].weight;
```

Setelah seluruh IV diiterasi, dilakukan qsort kemudian dilakukan pengecekan sekaligus dipilih byte yang memiliki probabilitas yang tinggi

```
while(!validchars[i][table[i][j].b]) {
  j++;
}
```

Kemudian di assign ke fullkeybuf

```
fullkeybuf[i+3] = table[i][j].b;
```

dari full keybuf tersebut kemudian di lakukan testing untuk mendekripsi.

Pada fungsi:

```
static void guesskeybytes(int ivlen, uint8_t * iv, uint8_t * keystream, uint8_t * result, int kb)
```

dilakukan proses menebak byte kunci.

REFERENSI

[1] http://www.item.ntnu.no/_media/people/personalpages/phd/anton/kleins_and_ptw_attacks_on_wep.pdf, diakses pada 19 Mei 2016

[2] https://eprint.iacr.org/2007/120.pdf, diakses pada 19 Mei 2016

[3] http://security.stackexchange.com/questions/57970/klein-and-ptw-wep-attack, diakses pada 19 Mei 2016

[4] http://juggernaut.wikidot.com/aircrack-ptw, diakses pada 19 Mei 2016

[5] https://matthieu.io/dl/wifi-attacks-wep-wpa.pdf, diakses pada 19 Mei 2016

[6] http://www.dummies.com/how-to/content/understanding-wep-weaknesses.html, diakses pada 19 Mei 2016

[7] http://cage.ugent.be/~ls/website2007/abstracts/brusselscontactforum_Andreas_Klein.pdf, diakses pada 19 Mei 2016

[8] https://kalitutorials.wordpress.com/2014/07/10/wifi-hack-crack-wep-passwords-with-kali/, diakses pada 19 Mei 2016

[9] https://linuxconfig.org/how-to-crack-a-wireless-wep-key-using-air-crack, diakses pada 19 Mei 2016

[10] http://www.informit.com/articles/article.aspx?p=102230&seqNum=10, diakses pada 19 Mei 2016

[11] https://simple.wikipedia.org/wiki/Known-plaintext_attack, diakses pada 19 Mei 2016

[12] http://www.dartmouth.edu/~madory/RC4/wepexp.txt, diakses pada 19 Mei 2016

[13] http://www.tripwire.com/state-of-security/latest-security-news/attack-exploits-weaknesses-in-rc4-algorithm-to-reveal-encrypted-data/, diakses pada 19 Mei 2016

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Mei 2016

Ibrohim Kholilul Islam
13513090