IMPLEMENTASI VIGENERE CIPHER PADA AJAX(*ASYNCHRONOUS JAVASCRIPT AND XML*)

Nur Cahya Pribadi – NIM : 13505062

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if15062@students.if.itb.ac.id

Abstrak

Perkembangan dunia internet dari hari ke hari semakin maju. Hal ini ditandai dengan bertambahnya pengguna internet dan ketergantungan orang kepada internet. Internet digunakan untuk menampilkan halaman *web*, *chatting*, *online gaming*, dan lain sebagainya. Seiring dengan perkembangan internet ini, perkembangan *web* pun semakin maju. Salah satu teknik pengembangan *web* yang baru populer saat ini adalah AJAX (*Asynchronous javascript and XML*).

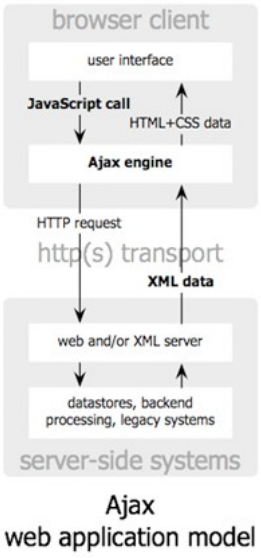
AJAX (*Asynchronous javascript and XML*) adalah suatu teknik pemograman berbasis *web* untuk menciptakan aplikasi *web* secara interaktif. AJAX digunakan agar sebuah aplikasi *web* dapat mengambil data atau informasi dari *server* secara tidak sinkronis tanpa merubah tampilan dan perilaku pada halaman yang ada.

Penggunaan AJAX dilakukan dengan *request* dari *client* terhadap *server* secara tidak sinkronis, yang kemudian *server* menanggapi *request* tersebut dengan mengirimkan *response*. Keamanan dalam penggunaan AJAX tersebut masih minimal, terutama digunakan untuk menerima informasi rahasia seperti nomor kartu kredit, kata kunci, dan lain sebagainya. Oleh karena itu diperlukan pengenkripsian pada *request* dari *client* maupun *response* dari *server*, sehingga informasi tersebut tidak dapat dibaca oleh orang yang tidak berhak. Pada pengenkripsian pada AJAX ini menggunakan algoritma Vigenère Cipher, karena algoritma ini mudah dimengerti dan dijalankan serta sulit dipecahkan untuk orang awam.

***Kata kunci**: Web,* enkripsi, Vigenère Cipher, AJAX.

1. Pendahuluan

Penggunaan AJAX dalam pembuatan *web* saat ini menjadi sangat penting. Dengan menggunakan AJAX mengurangi penggunaan *bandwidth* dan *load time, web* menjadi lebih interaktif dan cepat, dan mengurangi koneksi ke *server*. Tapi penggunaan AJAX ini tidak menjamin keamanan pada *web* tersebut meningkat.



Gambar 1. Model aplikasi *web* AJAX

Gambar 1 menunjukkan model koneksi antara *client* (*browser client*) dengan *server* (*server-side system*) dengan menggunakan AJAX. Pada gambar tersebut *client* mengirimkan sebuah *request* berupa HTTP *request* kepada server, biasanya HTTP *request* ini berisikan jenis informasi yang diminta *client* kepada server. Setelah itu server menerima *request* kemudian mengirimkan data atau informasi berupa XML data sesuai dengan *request* yang diminta.

2. Vigenère Cipher

Vigenère Cipher adalah metode menyandikan teks alfabet dengan menggunakan deretan sandi Caesar berdasarkan huruf-huruf pada kata kunci. Vigenère Cipher merupakan bentuk sederhana dari sandi substitusi polialfabetik. Kelebihan sandi ini dibanding sandi Caesar dan sandi monoalfabetik lainnya adalah sandi ini tidak begitu rentan terhadap metode pemecahan sandi yang disebut analisis frekuensi. Vigenère Cipher merupakan cipher abjad majemuk yang paling dikenal karena menghasilkan banyak varian atau *cipher* turunan seperti Beaufort, Gronsfeld, Porta, dan sebagainya.

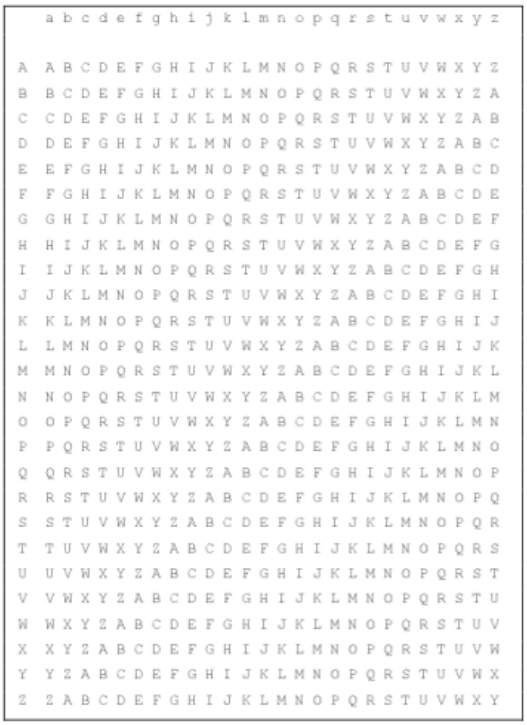
Giovan Batista Belaso menjelaskan metode ini dalam buku La cifra del. Sig. Giovan Batista Belaso (1553); dan disempurnakan oleh diplomat Perancis Blaise de Vigenère, pada 1586. Vigenère menuliskan hasil penemuannya dalam buku/traktat yang berjudul "*Traicte des Chiffres*" Pada abad ke-19, banyak orang yang mengira Vigenère adalah penemu sandi ini, sehingga, sandi ini dikenal luas sebagai "Vigenère Cipher".

Sandi ini dikenal luas karena cara kerjanya mudah dimengerti dan dijalankan, dan bagi para pemula sulit dipecahkan. Pada saat kejayaannya, sandi ini dijuluki *le chiffre indéchiffrable* (bahasa Prancis: 'sandi yang tak terpecahkan'). Metode pemecahan sandi ini baru ditemukan pada abad ke-19. Pada tahun 1854, Charles Babbage menemukan cara untuk memecahkan sandi Vigenère. Metode ini dinamakan tes Kasiski karena Friedrich Kasiski-lah yang pertama mempublikasikannya.

Vigènere Cipher digunakan oleh tentara konfederasi (*Confederate Army*) pada Perang Sipil Amerika (American Civil War). Perang Sipil terjadi setelah Vigènere Cipher berhasil dipecahkan. Hal ini diilustrasikan oleh kutipan pernyataan Jenderal Ulysess S. Grant: "*It would sometimes take too log to make translation of intercepted dispatches for us to receive any benefit from them, but sometimes they gave useful information*".

Sandi Vigenère sebenarnya merupakan pengembangan dari sandi Caesar. Pada sandi Caesar, setiap huruf teks terang digantikan dengan huruf lain yang memiliki perbedaan tertentu pada urutan alfabet. Misalnya pada sandi Caesar dengan geseran 3, A menjadi D, B menjadi E and dan seterusnya. Sandi Vigenère terdiri dari beberapa sandi Caesar dengan nilai geseran yang berbeda.

Untuk menyandikan suatu pesan, digunakan sebuah tabel alfabet yang disebut tabel Vigenère. Tabel Vigenère berisi alfabet yang dituliskan dalam 26 baris, masing-masing baris digeser satu urutan ke kiri dari baris sebelumnya, membentuk ke-26 kemungkinan sandi Caesar. Setiap huruf disandikan dengan menggunakan baris yang berbeda-beda, sesuai kata kunci yang diulang



Gambar 2. Bujur Sangkar Vigenère

Misalnya, teks terang yang hendak disandikan adalah perintah "SERBU BERLIN", sedangkan

kata kunci antara pengirim dan tujuan adalah "pizza". Huruf pertama pada plainteks, S, disandikan dengan menggunakan baris berjudul P, huruf pertama pada kata kunci. Pada baris P dan kolom S di tabel Vigenère, terdapat huruf H. Demikian pula untuk huruf kedua, digunakan huruf yang terletak pada baris I (huruf kedua kata kunci) dan kolom E (huruf kedua plainteks), yaitu huruf M. Proses ini dijalankan terus sehingga akan didapatkan :

Plainteks : SERBUBERLIN
Kunci : PIZZAPIZZAP
Cipherteks : HMQAUQMQKIC

Proses sebalinya (disebut dekripsi), dilakukan dengan mencari huruf teks bersandi pada baris berjudul huruf dari kata kunci. Misalnya, pada contoh diatas, untuk huruf pertama, kita mencari huruf H (huruf pertama teks tersandi) pada baris P (huruf pertama pada kata kunci), yang terdapat pada kolom S, sehingga huruf pertama adalah S. Lalu M terdapat pada baris I di kolom E, sehingga diketahui huruf kedua teks terang adalah E, dan seterusnya hingga didapat perintah "serbuberlin". Bila dicermati, setiap huruf hasil enkripsi pada Vigenère Cipher merupakan Caesar Cipher dengan kunci yang berbeda-beda. Contoh:
C(‘C’) = (‘C’ + ‘t’) mod 26 = V
C(‘T’) = (‘T’ + ‘e’) mod 26 = X

3. Pencurian Informasi yang Dapat Terjadi pada Penggunaan AJAX

Ada 2 macam pencurian informasi yang mungkin dapat terjadi dalam penggunaan AJAX yaitu pencurian informasi dengan mengetahui pola *request client* dan pencurian informasi dengan *sniffing*.

3.1 Mencuri Informasi dengan Mengetahui Pola *Request Client*

```
function requestInfo(name){
        httpReqGetDetailCA.open('get',
'info.php?
name='+name+'&nocache='+Math.random());

httpReqGetDetailCA.onreadystatechange =
handleGetDetailCA; //fungsi utk menangani
respon server
        httpReqGetDetailCA.send(null);
}
```

Pada kode diatas merepresentasikan pengiriman *request* dari *client* ke server. *Request* yang dikirimkan berupa text dengan variabel “name” dan ingin meminta *response* dari server berupa informasi seseorang yang mempunyai nama sesuai variabel “name” tersebut. Jika seorang hacker melihat kode javascript ini dan mengetahui pola *request* yang dikirimkan oleh *client*, maka dengan sangat mudah hacker mengetahui informasi orang lain dengan menebak nama pada *request*. Oleh karena itu dibutuhkan pengenkripsian pada pengiriman *request* sehingga peluang hacker untuk menebak informasi *request* menjadi semakin sulit.

3.2 Mencuri Informasi dengan *Sniffing*

Sniffing adalah melakukan pencurian/penyadap paket pada jaringan komputer. Untuk melakukan *sniffing* ini dibutuhkan *sniffer*. *Sniffer* adalah sebuah aplikasi yang dapat melihat lalu lintas data pada jaringan komputer. Dikarenakan data mengalir secara bolak-balik pada jaringan, aplikasi ini menangkap tiap-tiap paket terkadang menguraikan isi dari RFC (*request for comment*) atau spesifikasi lain. Berdasarkan pada struktur jaringan (hub atau switch), salah satu pihak dapat menyadap keseluruhan atau salah satu *dari pembagian lalu lintas dari salah satu mesin di jaringan*. P*erangkat pengendali jaringan dapat pula diatur oleh aplikasi penyadap untuk bekerja dalam promiscuous mode untuk mendengarkan semuanya.*

Sniffer paket dapat dimanfaatkan untuk hal-hal berikut:

- *Mengatasi permasalahan pada jaringan komputer.*
- *Mendeteksi adanya penyelundup dalam jaringan (Network Intusion).*
- *Memonitor penggunaan jaringan dan menyaring isi isi tertentu.*
- *Memata-matai pengguna jaringan lain dan mengumpulkan informasi pribadi yang dimilikinya (misalkan password).*
- *Dapat digunakan untuk Reverse Engineer pada jaringan.*

Pada AJAX ini tentu saja sniffing digunakan untuk menyadap isi request dan response. Penyadapan ini dapat dilakukan pada request

dan response.Oleh karena itu diperlukan pengenkripsian pada request maupun response.

4. Implementasi Vigenère Cipher pada *Browser Client*

Pada *browser client* pengimplementasian Vigenère Cipher hanya dapat dilakukan dengan bahasa javascript. Untuk pengenkripsian dan pendekripsian *plaintext* pada Vigenère Cipher dibutuhkan bujur sangkar Vigenère dengan kode sebagai berikut :

```
var bs = new Array();
for(i=0;i<26;i++){
        bs[i] = new Array();
        for(j=0;j<26;j++){
                bs[i][j]=(i+j)%26;
        }
}
```

Diawali dengan membuat sebuah variabel global dengan nama "bs" dengan tipe array of array. Pada bs[i][j], i merpresentasikan a-z untuk kunci dan j merepresentasikan a-z untuk *plaintext*. Untuk mempermudah pembuatan bujur sangkar Vigenère maka semua huruf diganti menjadi angka. Huruf "a" diganti menjadi 0, huruf "b" diganti menajdi 1, dan demikian seterusnya sehingga huruf "z" digantikan menjadi 25.

Rangkaian kode diatas terdapat 2 kali pengulangan. Pengulangan pertama merupakan pengulangan dari 0 sampai 25 untuk kunci.

```
for(i=0;i<26;i++)
```

Pengulangan kedua merupakan pengulangan dari 0 sampai 25 untuk *plaintext*.

```
for(j=0;j<26;j++)
```

Oleh karena itu jumlah nilai yang tersimpan pada "bs" adalah 26*26 nilai. Masing-masing dari nilai tersebut adalah hasil modulus 26 dari pertambahan i dan j.

```
bs[i][j]=(i+j)%26;
```

Selain dibutuhkan bujur sangkar Vigenère dibutuhkan pula sebuah variabel global untuk merepresentasikan kunci untuk Vigenère Cipher.

```
var key = "vigenere";
```

4.1 Enkripsi dengan Vigenère Cipher pada *Browser Client*

```
function encrypt(text){
  var binchar, bin="",l,ikey,ik;
  for (k = 0; k < text.length; k++) {
                if(text.charCodeAt(k)>96     &&
text.charCodeAt(k)<123){
     ikey=k%(key.length);
     ik = key.charCodeAt(ikey)-97;
     l = text.charCodeAt(k)-97;
     binchar = toChar(bs[ik][l]);
   }else{
     binchar = text.charAt(k);
   }
   bin += binchar;
  }
  return bin;
}
```

Fungsi enkripsi ini menerima masukan berupa string dengan menggunakan variabel "text". Dari "text" tersebut diambil karakter per karakter untuk dienkripsikan.

```
  for (k = 0; k < text.length; k++)
```

Kemudian dilakukan validasi apakah karakter masih dalam lingkup bilangan ascii atau tidak. Ketika tidak masuk dalam validasi maka karakter tidak dienkripsi.

```
binchar = text.charAt(k);
```

Sebaliknya jika masuk dalam validasi maka akan Dilakukan pencarian karakter kunci yang nantinya akan mengenkripsi karakter yang telah diambil dari "text". Kemudian dari 2 karakter yang didapat(karakter dari kunci dan karakter

dari “text”) di *convert* menjadi ke bilangan ascii dan masing-masing dikurangi 97(97 merupakan nilai bilangan ascii “a”) agar nilai tersebut berada pada range 0-25 sehingga cocok dengan bujur sangkar Vigenère “bs”.

```
ikey=k%(key.length);
ik = key.charCodeAt(ikey)-97;
l = text.charCodeAt(k)-97;
```

Kemudian dari 2 nilai bilangan tersebut di cocokkan dengan bujur sangkar Vigenère “bs” dan akan mendapatkan nilai pada range 0-25 yang kemudian nilai tersbut di-*convert* menjadi sebuah karakter baru yaitu karakter yang terenkripsi.

```
binchar = toChar(bs[ik][l]);
```

Dari masing-masing karakter yang telah terenkripsi digabungkan sehingga membentuk sebuah string yang utuh.

```
bin += binchar;
```

4.2 Dekripsi dengan Vigenère Cipher pada *Browser Client*

```
function decrypt(text){
  var binchar, bin="",found,ikey,ik;
  for (k = 0; k < text.length; k++) {
                if(text.charCodeAt(k)>96     &&
text.charCodeAt(k)<123){
     ikey=k%(key.length);
     ik = key.charCodeAt(ikey)-97;
     l = text.charCodeAt(k)-97;
     found = null;
     var h=0;
     while(h<26 && !found){
       if(bs[ik][h]==l)
       found = h;
       h++;
     }
     binchar = toChar(found);
   }else{
     binchar = text.charAt(k);
   }
   bin += binchar;
  }
  return bin;
}
```

Secara sekilas pada kode dekripsi ini hampir sama dengan kode enkripsi seperti melakukan pengambilan karakter per karakter pada “text”(pada enkripsi sebagai *plaintext*, sedangkan pada dekripsi sebagai *ciphertext*), mencari karakter kunci, melakukan *convert* 2 karakter (karakter dari kunci dan karakter dari “text”). Hanya saja pada dekripsi ini berusaha untuk menemukan karakter *plaintext* pada bujur sangkar “bs” dengan menggunkan karakter kunci dan karakter dari “text” (*ciphertext*) yang telah diketahui sebelumnya.

```
found = null;
var h=0;
while(h<26 && !found){
  if(bs[ik][h]==l)
      found = h;
  h++;
}
```

5. Implementasi Vigenère Cipher pada *Server*

Pengimplementasian Vigenère Cipher pada *server* tidak jauh beda dengan impelmentasi pada *browser client*, hanya saja pada *server* ini pengimplementasian menggunakan bahasa php sebagai bahasa pada *server*. Pengimpelementasian pada *server* ini juga membutuhkan bujur sangkar Vigenère dan kunci global.

```
$key = "vigenere";
$bs = array();
for($i=0;$i<26;$i++){
        $bs[$i] = array();
        for($j=0;$j<26;$j++){
                $bs[$i][$j]=($i+$j)%26;
```

```
        }
}
```

5.1 Enkripsi dengan Vigenère Cipher pada *Server*

```
function encrypt($text){
  global $key, $bs;
  for ($k = 0; $k < strlen($text); $k++) {
                    if(ord($text[$k])>96      &&
ord($text[$k])<123){
     $ikey=$k%(strlen($key));
     $l = ord($text[$k])-97;
     $ik = ord($key[$ikey])-97;
     $binchar = toChar($bs[$ik][$l]);
    }else{
     $binchar = $text[$k];
    }
    $bin .= $binchar;
  }
  return $bin;
}
```

Secara skema kode, implementasi enkripsi pada *server* sama dengan implementasi pada *browser client*. Perbedaannya hanya pada sintaks pemogramannya karena pada *server* menggunakan bahasa PHP sedangkan pada *browser client* menggunakan bahasa javascript.

5.2 Dekripsi dengan Vigenère Cipher pada *Server*

```
function decrypt($text){
  global $key, $bs;
  for ($k = 0; $k < strlen($text); $k++) {
                    if(ord($text[$k])>96      &&
ord($text[$k])<123){
     $ikey=$k%(strlen($key));
     $ik = ord($key[$ikey])-97;
     $l = ord($text[$k])-97;
     $found = null;
     $h=0;
     while($h<26 && !$found){
      if($bs[$ik][$h]==$l)
       $found = $h;
      $h++;
     }
     $binchar = toChar($found);
    }else{
     $binchar = $text[$k];
    }
    $bin .= $binchar;
  }
  return $bin;
}
```

Secara skema kode, implementasi dekripsi pada *server* sama dengan implementasi pada *browser client*. Perbedaannya hanya pada sintaks pemogramannya karena pada *server* menggunakan bahasa PHP sedangkan pada *browser client* menggunakan bahasa javascript.

6. Impelentasi AJAX dengan Menggunakan Enkripsi dan Dekripsi pada *Browser Client* dan *Server*

Browser client mengirimkan *request* ke *server* dengan *request* telah terenkripsi terlebih dahulu.

```
//konfigurasi http request
isMSIE = (navigator.appName=="Microsoft
Internet Explorer");
if (isMSIE) {//khusus browser IE
                    httpReq = new
ActiveXObject("Microsoft.XMLHTTP");
}else{
  httpReq = new XMLHttpRequest();
}
```

Pada kode diatas mengkonfigurasi http *request*. Penggunaan http *request* pada *browser* Internet Explorer berbeda dengan *browser* lainnya seperti Mozilla Firefox, Opera, dan lain-lain.

```
function getInfo(name){
        httpReq.open('get', 'ajax.php?
name='+encrypt(name)
+'&nocache='+Math.random());
        httpReq.onreadystatechange =
handleGetInfo; //fungsi utk menangani respon
server
        httpReq.send(null);
}
```

Pada kode diatas, *client* mengirimkan *request* ke *server* dengan menggunakan metode ”get” melalui variabel ”name” dengan nilai **encrypt(name)** dengan name berupa input fungsi. Kemudian panggil fungsi getInfo tersebut.

```
getInfo("adi");
```

Pada contoh diatas *client* mengirim *request* “vlo” (hasil enkripsi dari “adi”). Setelah *client* mengirimkan *request*, *server* mendekripsikan *request* tersebut dan mengirimkan *response* sesuai dengan *request* yang diterima.

```
$name = $_GET["name"];
if(decrypt($name)=="adi"){
        echo encrypt("nama lengkapnya adi
sibutar-butar");
}
```

Server mengirimkan *response* “iise pvrbsgtacr vlo fmsyoix-oykem” (hasil enkripsi dari “nama lengkapnya adi sibutar-butar”) ke *browser*. Dan di *client response* tersebut didekripsi.

```
function handleGetInfo(){
    if ((httpReq.readyState == 4) &&
(httpReq.status == 200)) { //jika respon selesai
   var response = httpReq.responseText;
   if (response) {        //jika respon berisi
sesuatu, update tampilan
     alert(decrypt(response));
   }
  }
}
```

7. Analisis Implementasi Vigenère Cipher pada AJAX

Kelebihan dari implementasi Vigenère Cipher ini pada ajax adalah

1. Mudah untuk diimplementasikan.
2. Panjang *plaintext* sama dengan panjang *ciphertext* sehingga tidak menggunakan *bandwidth* lebih.
3. Komputasi tidak terlalu banyak sehingga tidak *overtime*.

Kekurangan dari implementasi Vigenère Cipher ini pada ajax adalah

1. Enkripsi dari Vigenère Cipher masih dapat dipecahkan walaupun sulit untuk orang awam.
2. Vigenère Cipher ini menggunakan kunci sehingga pada *browser client* masih dapat dilihat kunci tersebut.
3. Perlu banyak *memory space* untuk menyimpan nilai-nilai pada bujur sangkar Vigenère.

8. Kesimpulan

Kesimpulan yang dapat dimbil dari implementasi Vigenère Cipher pada AJAX ini adalah:

1. AJAX dapat mengurangi penggunaan *bandwidth* dan *load time*, *web* menjadi lebih interaktif dan cepat, dan mengurangi koneksi ke *server*. Tapi penggunaan AJAX ini tidak menjamin keamanan pada *web* tersebut meningkat.
2. Vigenère Cipher dikenal luas karena cara kerjanya mudah dimengerti dan dijalankan, dan bagi para pemula sulit dipecahkan.
3. Ada 2 macam pencurian informasi yang mungkin dapat terjadi dalam penggunaan AJAX yaitu pencurian informasi dengan mengetahui pola *request client* dan pencurian informasi dengan *sniffing*.
4. Kelebihan dari implementasi Vigenère Cipher ini pada ajax adalah mudah untuk diimplementasikan, panjang *plaintext* sama dengan panjang

ciphertext, komputasi tidak terlalu banyak.

5. Kekurangan dari implementasi Vigenère Cipher ini pada ajax adalah enkripsi dari Vigenère Cipher masih dapat dipecahkan walaupun sulit untuk orang awam, *browser client* masih dapat dilihat kunci, perlu banyak *memory space* untuk menyimpan nilai-nilai pada bujur sangkar Vigenère.

DAFTAR PUSTAKA

[1] Munir, Rinaldi. (2004). Bahan Kuliah IF3058 Kriptografi. Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

[2] Wrox Article: programmer to programmer. *http://www.wrox.com/WileyCDA/Section/id-303217.html*. Tanggal akses: 2 April 2009 pukul 20:00.

[3] Mozilla Developer Center. *https://developer.mozilla.org/En/AJAX:Getting_Started*. Tanggal akses: 2 April 2009 pukul 20:10.

[4] Adaptive Path. *http://www.adaptivepath.com/ideas/essays/archives/000385.php*. Tanggal akses: 2 April 2009 pukul 20:20.

[5] Karisma, Jimmy. (2008). Analisis Enkripsi Halaman *Web* Berbasis HTML dengan Menggunakan Vigenère Cipher. . Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

[6] PHP:Hypertext Preprocessor. http://php.net/. Tanggal akses: 2 April 2009 pukul 20:20.

[7] W3Schools Online Web Tutorials. http://www.w3schools.com/. Tanggal akses: 2 April 2009 pukul 20:20.