Studi dan Implementasi Penggunaan Enkripsi Hash dan Tanda Tangan Digital untuk Pengamanan *Password* dari *Client* ke *Web Server*

Charles Hariyadi (13505105)[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*if15105@students.if.itb.ac.id(charles@charles.web.id)*

ABSTRAK

Saat ini situs *web* (*web site*) telah mengalami perkembangan yang pesat, sehingga banyak dimanfaatkan untuk keperluan yang beragam. *Web* saat ini bukan hanya digunakan sebagai sarana untuk tukar menukar informasi, *web* bahkan telah menjadi salah satu sarana komunikasi social (*facebook*), sarana untuk tukar menukar file (*file sharing*), sarana bisnis (*e-commerce*) dan keperluan lainnya. Hal tersebut menjadikan *web* sebagai tempat penyimpanan informasi-informasi penting bagi para penggunanya. Untuk setiap *web* tersebut biasanya diperlukan suatu proses autentikasi untuk menjaga kerahasiaan data pengguna. Autentikasi biasanya dilakukan dengan melakukan pencocokan nama user yang terdaftar beserta sandi lewat yang telah terdaftar atas nama pengguna tersebut. Karena informasi ini bersifat penting, diperlukan suatu proses penyimpanan sandi lewat yang lebih aman dan terpercaya.
"Aman" dan "terpercaya" dalam hal ini merupakan suatu mekanisme penyimpanan dan pengelolaan informasi sandi lewat yang akan digunakan oleh pengguna. Biasanya sandi lewat disimpan pada basis data dalam bentuk teks biasa. Sehingga terdapat kemungkinan terjadi penyalahgunaan oleh oknum-oknum yang bertanggung jawab dalam pengelolaan basis data tersebut. Pendekatan lain yang dilakukan adalah dengan menggunakan *cipher* yang banyak digunakan saat ini seperti *Data Encryption Standard* (DES) atau RC2. Pendekatan ini biasanya merupakan pendekatan yang cukup aman, tetapi jika terdapat seseorang yang mengetahui kata kunci yang digunakan, maka sandi lewat dapat dengan mudah dipecahkan. Terdapat juga kemungkinan lain yaitu penyadapan yang dilakukan oleh perangkat penyadap jaringan untuk mendapatkan informasi penting itu.
Karena itulah makalah ini akan mencoba melakukan studi dan implementasi enkripsi terhadap sandi lewat menggunakan fungsi *hash* dan juga tanda tangan digital untuk memastikan keamanan dalam pengiriman dan pengelolaan sandi lewat. Diharapkan hasil dari pengembangan ini dapat memberikan kontribusi terhadap pengamanan sandi lewat pada komunikasi dari *client* ke sebuah situs *web*.

Kata kunci : *password*, hash, tanda tangan digital, *client server*.

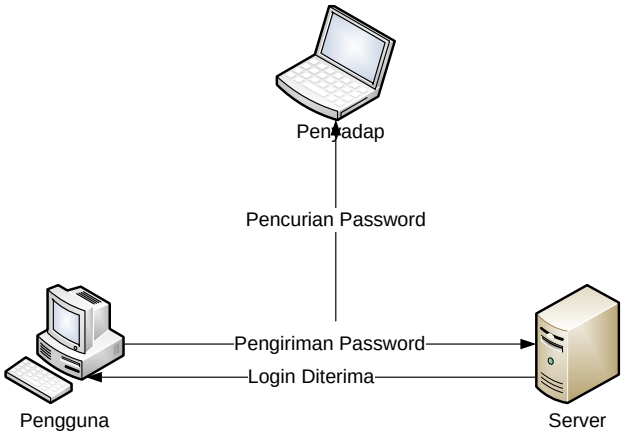
I. PENDAHULUAN

Semua orang selalu menginginkan keamanan dalam menjalankan setiap aktivitasnya. Demikian halnya dalam dunia maya, pengaksesan terhadap suatu situs juga membutuhkan penanganan terhadap keamanan bagi penggunanya. Keamanan terhadap pengguna situs biasanya dilakukan melalui pemanfaatan sandi lewat untuk pengaksesan terhadap informasi-informasi yang ada pada situs. Dengan adanya *password* (sandi lewat), seorang pengguna dapat dengan leluasa mengakses informasi yang disediakan oleh situs tersebut ataupun informasi-informasi yang dimasukkannya ke situs tersebut. Tetapi saat ini sandi lewat tidak lagi memberikan keamanan sempurna, karena sandi lewat rentan terhadap serangan-serangan.

Dewasa ini terdapat berbagai serangan yang dapat dilakukan oleh pihak-pihak tidak bertanggung jawab terhadap sandi lewat yang digunakan dalam suatu situs. Serangan-serangan itu dapat berupa penyerangan menggunakan perangkat lunak jaringan untuk mengambil informasi sandi lewat yang dikirimkan oleh pengguna situs ketika melakukan *login* ke suatu situs (gambar 1), penyalahgunaan kuasa oleh pihak yang mengelola basis data pengguna, dan serangan-serangan lainnya. Karena itulah diperlukan suatu metode yang "Aman" dan "Terpercaya" dalam penanganan sandi lewat yang merupakan informasi berharga ini.

Sebuah situs biasanya dibangun menggunakan teknologi *Hypertext Markup Language* (HTML), dan biasanya hubungan antara pengguna dengan situs dijembatani oleh teknologi PHP *Hypertext Preprocessor*

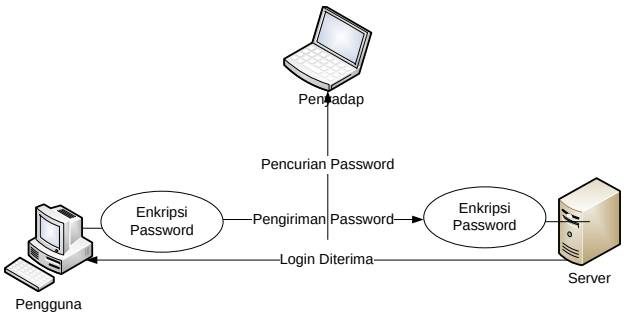
(PHP). PHP merupakan sebuah teknologi yang membantu hubungan antara *client* dan *server*,dengan adanya teknologi ini, suatu situs dapat menyediakan layanan yang lebih menarik dan aman. Untuk masalah *login* misalnya, PHP dapat menyediakan fungsi *login* yang memungkinkan penggunanya untuk melakukan *login* terhadap suatu situs dan melakukan autentikasi terhadap keberadaan sang pengguna pada situs. Sehingga pengguna dapat menggunakan fasilitas yang disediakan oleh situs selama pengguna masih *logged in* dalam situs tersebut. Tetapi dalam penerapannya, masih terdapat beberapa kelemahan mekanisme keamanan yang biasanya digunakan pada penggunaan PHP. Salah satu kelemahan itu adalah kelemahan dalam penanganan informasi sandi lewat.



Gambar 1 Pencurian Sandi Lewat Oleh Pihak ketiga

Informasi sandi lewat biasanya dikirimkan dalam bentuk teks dan tidak terenkripsi. Hal ini memudahkan pengambilan informasi tersebut baik melalui penyadapan pada sisi *client* maupun penyalahgunaan informasi pada sisi *server* (dalam hal basis data). Informasi sandi lewat berupa teks merupakan salah satu penanganan sandi lewat terlemah, jika terjadi penyadapan informasi maka penyerang akan dengan mudah mengetahui informasi sandi lewat yang digunakan. Pada basis data, informasi sandi lewat berupa teks memungkinkan seorang oknum yang memiliki hak khusus terhadap basis data melakukan pengaksesan terhadap informasi sandi lewat yang seharusnya bersifat privat. Karena itu diperlukan enkripsi terhadap pengiriman dan penyimpanan data sehingga tidak terjadi penyalahgunaan informasi sandi lewat oleh pihak yang tidak berwenang.

Saat ini telah tersedia berbagai jenis teknik enkripsi yang dapat digunakan untuk mengamankan informasi sandi lewat. Teknik enkripsi untuk pengamanan sandi lewat biasanya datang dari teknik enkripsi hash. Biasanya teknik enkripsi ini diterapkan pada saat seorang pengguna melakukan pendaftaran akun ke sebuah situs ataupun pada saat pengguna melakukan *login* ke situs tersebut. Pada saat akan melakukan *login* ke suatu situs, pengguna akan memasukkan sandi lewat dan nama pengguna (kondisi normal), kemudian informasi ini akan dienkripsi terlebih dahulu sebelum dikirimkan ke server untuk diproses oleh *server* (Gambar 2). Setelah itu pada saat informasi tiba di *server*, informasi akan terlebih dahulu didekripsi untuk selanjutnya dicocokkan dengan informasi di basis data. Biasanya informasi di basis data disimpan dalam bentuk teks terenkripsi hasil enkripsi pada saat pendaftaran. Jika pencocokan berhasil, maka sang pengguna dapat melakukan login ke situs. Dengan begitu kemanan informasi sandi lewat dapat terjamin.



Gambar 2 **Proses Enkripsi dan Dekripsi Sandi Lewat Pada Hubungan *Client-Server***

Makalah ini akan membahas mengenai implementasi teknik hash dan tanda tangan digital pada pengolahan sandi lewat. Penerapan kedua teknik enkripsi ini dimaksudkan untuk mengatasi masalah-masalah keamanan yang mungkin muncul pada pemrosesan sandi lewat. Diharapkan makalah ini mampu memberikan suatu metode pengamanan yang lebih terpercaya dalam pemrosesan sandi lewat.

II. ALGORITMA ENKRIPSI

II.1 SHA-1

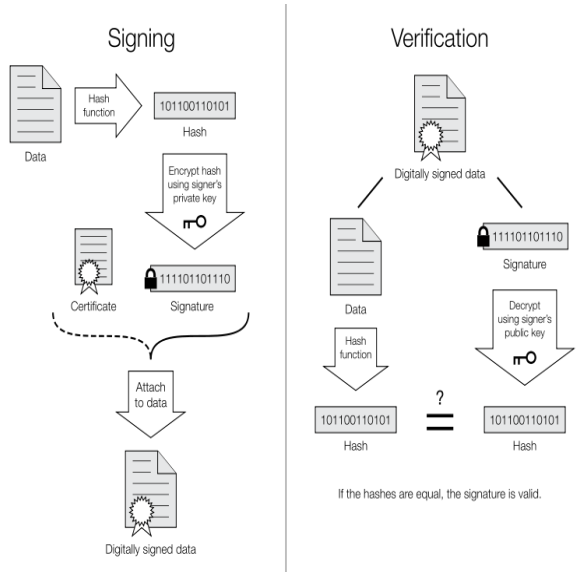
SHA-1 digest adalah fungsi hash satu-arah yang berfungsi untuk menjadi sebuah kunci khusus pada sebuah dokumen. SHA-1 akan membangkitkan sebuah kunci khusus yang nantinya berguna untuk menjadi pembanding keaslian dari sebuah dokumen. SHA-1 merupakan algoritma yang tidak jauh berbeda dari MD4. Algoritma SHA-1 menerima masukan berupa pesan dengan ukuran sembarang (sebesar 6^64 bits) dan menghasilkan message digest yang panjangnya 160 bit. Langkah-langkah pembuatan message digest secara garis besar:

[1] Penambahan bit-bit pengganjal (padding bits).
[2] Penambahan nilai panjang pesan semula.
[3] Inisialisasi penyangga (buffer) SHA
[4] Pengolahan pesan dalam blok berukuran 512 bit

Belum ada penyerangan terhadap SHA karena panjang

bit hash yang dihasilkan, 160 bit. Dan SHA merupakan algoritma yang lebih kuat terhadap serangan brute-force ataupun birthday-attack, dari pada MD5 yang memiliki panjang 128bit. Tidak ada dekripsi untuk algoritma jenis ini, hal ini dikarenakan *message digest* (pesan hasil enkripsi) yang dihasilkan tidak dapat lagi di terjemahin menjadi informasi pesan yang dimasukkan di awal.

II.2 Digital Signature



***Gambar 3* Proses Penandatanganan Digital**

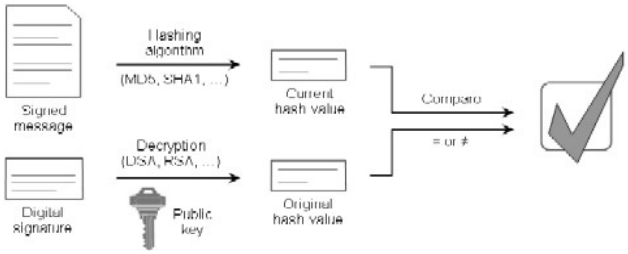
Digital signature (tanda tangan digital) merupakan proses pemberian tanda tangan pada sebuah dokumen atau data yang nantinya akan digunakan untuk melakukan verifiksi terhadap integritas dari dokumen. Tanda tangan digital memanfaatkan 2 kunci utama :

1. Kunci Publik yang merupakan sebuah nilai yang digunakan pada saat pengguna menggunakan pembubuhan tanda tangan digital pada data. Kunci ini untuk kemudian akan diverifikasikan menggunakan kunci publik yang dimiliki oleh pemilik (pemilik situs). Kunci ini tidak bersifat rahasiadan biasanya disebarkan kepada orang-orang yang akan melakukan komunikasi dengan *server*.

2. Kunci Private merupakan sebuah nilai yang hanya dimiliki khusus oleh pemilik dari tanda tangan digital. Dengan kunci private, seseorang dapat dengan bebas melakukan penandaan dan dekripsi terhadap dokumen yang ditandai dengan kunci publik yang berhubungan dengan kunci private ini. Apabila sebuah kunci private diketahui oleh pihak lain yang bukan pemilik, maka kunci private ini menjadi tidak berguna. Karena dengan mengetahui nilai dari kunci private ini, seseorang dapat melakukan pendekripsian terhadap semua dokumen atau data yang berhubungan dengan kunci private ini. Jika hal itu terjadi, harus diadakan pengubahan kunci private dan kunci publik yang berkoresponden dengannya.

Proses verifikasi terhadap tanda tangan digital dilakukan melalui beberapa tahap sebagai berikut :

1. Tahap pertama yaitu melakukan kalkulasi terhadap nilai hash yang dimiliki saat ini, ini dilakukan dengan melakukan hash terhadap pesan yang dimasukkan saat ini (dalam hal ini pesan berupa sandi lewat).
2. Tahap kedua yaitu melakukan kalkulasi terhadap nilai hash sebenarnya dengan melakukan dekripsi terhadap tanda tangan digital menggunakan kunci publik yang ada. Sehingga didapatkan nilai hash sebenarnya dari pesan (dalam hal ini berupa sandi lewat yang tersimpan dalam basis data).
3. Tahap ketiga merupakan tahap pencocokan nilai hash saat ini dengan nilai hash sebenarnya (hasil dari tahap 1 dan 2). Jika nilai keduanya sama maka pesan dinyatakan valid, jika nilai keduanya berbeda maka pesan dinyatakan tidak valid.



***Gambar 4* Proses Verifikasi Tanda Tangan Digital**

III. IMPLEMENTASI

III.1.Implementasi SHA

Terdapat banyak cara untuk meimplementasikan SHA dalam PHP, melalui *library* yang disediakan oleh pihak ketiga maupun fungsi yang disediakan oleh PHP sendiri. Dalam makalah ini penulis akan menggunakan fungsi SHA yang telah disediakan oleh PHP, mengingat fungsi SHA yang disediakan oleh PHP merupakan fungsi yang mudah untuk diterapkan dan juga efisien.

```
<?php
/****Registrasi *****/
/*
bagian program untuk melakukan sha
pada saat registrasi pengguna
dilakukan SHA pada sandi lewat
pengguna
*/

function SaveShaPassword()
{
    if(isset($_POST['password']))
```

```
      {
            $passwordHash =
sha1($_POST['password']);
            $sql = 'INSERT INTO user
(username,passwordHash) VALUES (?,?)';
            $result = $db->query($sql,
array($_POST['username'],
$passwordHash));
      }
  }

  ?>
```

Fungsi diatas merupakan fungsi yang akan digunakan pada saat registrasi dari pengguna, pada saat pengguna melakukan registrasi akan dilakukan pembangkitan nilai hash dari sandi lewat yang dimasukkan oleh pengguna untuk selanjutnya disimpan dibasis data. Fungsi untuk melakukan hash sha1 pada php dapat dilakukan dengan perintah sha1("kata sandi lewat"). Tapi karena tingkat keamanan sandi lewat yang hanya di hash dengan SHA rentan terhadap serangan, dilakukan modifikasi dengan menggunakan "***Salt***".

Salt merupakan sebuah *string*/kata tambahan yang biasanya ditambahkan sebelum pembangkitan nilai hash. *Salt* merupakan kumpulan dari karakter random dengan panjang tertentu dan biasanya ditambahkan ke *string* hasil hash. Nilai *Salt* ini untuk kemudian disimpan pada basis data ataupun disimpan secara explisit pada kode php dan digunakan pada saat melakukan verifikasi terhadap *login* dari pengguna. Berikut ini merupakan kode pembangkitan *Salt* yang dapat digunakan.

```
  <?php

  define('SALT_LENGTH', 9);

  function GenerateShaSalt($username,
$password, $salt = null)
  {
      if ($salt === null)
      {
          $salt =
substr(md5(uniqid(rand(), true)), 0,
SALT_LENGTH);
      }
      else
      {
          $salt = substr($salt, 0,
SALT_LENGTH);
      }
      $passwordHash =
$salt.sha1($salt, $password);
      /*kode ini bisa disertakan
disini untuk langsung melakukan
penyimpanan ke basis data*/
      /*
      $sql = 'INSERT INTO user
(username,passwordHash,salt) VALUES
(?,?,?)';
      $result = $db->query($sql,
array($username,
$passwordHash,$salt));
      */

      return $salt . sha1($salt .
$password);
  }

  ?>
```

Dengan adanya *Salt*, penyerangan terhadap sandi lewat akan lebih sulit karena penyerang harus dapat menebak nilai dari *Salt* selain melakukan penyerangan untuk mengetahui sandi lewat sebenarnya. Untuk melakukan pengecekan terhadap sandi lewat, dapat digunakan fungsi dibawah ini :

```
  <?php

  /* Nilai Salt telah diketahui */

  $salt = "89e495e4a";
  $passwordHash =
sha1($salt,$_POST['password']);
  $passwordHashReal =
$salt.$passwordHash;

  $sql = 'SELECT username FROM user
WHERE username = ? AND passwordHash =
?';
  $result = $db->query($sql,
array($_POST['username'],
$passwordHashReal));
  if ($result->numRows() < 1)
  {
      /* Tidak berhasil login */
      echo 'Maaf, informasi yang anda
masukkan salah!';
  }
  else
  {
      /* Login Berhasil */
      printf('Selamat datang %s!',
$_POST['username']);
  }

  ?>
```

III. 2 Implementasi Tanda Tangan Digital

Pada makalah ini, implementasi tanda tangan digital menggunakan fungsi *open source* yang bernama RSA1.3 dan dapat digunakan, didistribusikan dan diubah dengan persyaratan GNU GPL. Fungsi ini telah dimodifikasi sedemikian rupa oleh penulis sehingga dapat memenuhi segala kebutuhan tanda tangan digital yang diperlukan dalam menyelesaikan makalah ini. Berikut ini adalah beberapa fungsi utama yang digunakan dalam pembangkitan nilai kunci publik dan kunci private.

```
<?
function generate_keys ($show_debug=0){

      global $primes,  $maxprimes;
      while (empty($e) || empty($d)) {
            /* dalam kasus ini, nilai $p dan $q didapat dari bilangan prima, sehingga
            dilakukan pencarian 2 nilai prima terkecil dan dimasukkan ke variabel $p dan $q
            Kedua variabel ini harus memiliki nilai yang berbeda*/
          $p    =    $primes[mt_rand(0, $maxprimes)];
          while (empty($q) || ($p==$q)) $q = $primes[mt_rand(0,  $maxprimes)];
          //Nilai N yang dibangkitkan menggunakan perkalian $p dan $q
          $n = $p*$q;

          //$pi    dibutuhkan    untuk mengkalkulasi D dimana d merupakan nilai Dekripsi(Public) dan E merupakan nilai enkripsi(private)
          $pi = ($p - 1) * ($q - 1);

          // kunci public
          $e = tofindE($pi,  $p,  $q);

          // kunci private
          $d = extend($e, $pi);

          $keys = array ($n,  $e,  $d);
      }
      return $keys;
  }
  ?>
```

Fungsi diatas akan melakukan pembangkitan terhadap nilai kunci public dan kunci private. Dengan sebelumnya membangkitkan nilai *p* dan *q* dari nilai prima terkecil, kemudian menentukan nilai *N* yang merupakan hasil perkalian dari *p* dan *q*. Setelah itu dibangkitkan nilai *pi* untuk membangkitkan kunci publik dan kunci private. Setelah kunci ini dibangkitkan selanjutnyadibutuhkan fungsi untuk melakukan penandatanganan terhadap sandi lewat yang akan digunakan.

```
  <?
  function rsa_encrypt ($m,  $e,  $n) {
      $asci = array ();
      for ($i=0; $i<strlen($m); $i+=3) {
          $tmpasci="1";
          for ($h=0; $h<3; $h++) {
              if ($i+$h <strlen($m)) {
                  $tmpstr = ord (substr ($m,  $i+$h,  1)) - 30;
                  if (strlen($tmpstr) < 2) {
                      $tmpstr ="0".$tmpstr;
                  }
              } else {
                  break;
              }
              $tmpasci .=$tmpstr;
          }
          array_push($asci, $tmpasci."1");
      }

      //Semua    angka    dienkripsi menggunakan rumus : X = M^E (mod N)  = block ^E mod N <-- RSA1.3.php
      for  ($k=0;  $k<  count  ($asci); $k++) {
          $resultmod                  = powmod($asci[$k],  $e,  $n);
          $coded .= $resultmod." ";
      }
      return trim($coded);
  }

  function rsa_decrypt ($c,  $d,  $n) {
      //Menghilangkan nilai spasi dari string dan menambahkannya ke array
      $decryptarray = split(" ",  $c);
      for        ($u=0;        $u<count ($decryptarray); $u++) {
          if ($decryptarray[$u] == "") {

array_splice($decryptarray,  $u,  1);
          }
      }
      //Melakukan    dekripsi    dengan menggunakan rumus : X = M^D (mod N) = block ^E mod N
      Each number is then decrypted using the RSA formula: block ^D mod N
      for           ($u=0;           $u< count($decryptarray); $u++) {
          $resultmod                  = powmod($decryptarray[$u],  $d,  $n);
          //remove leading and trailing '1' digits
          $deencrypt.=           substr ($resultmod, 1, strlen($resultmod)-2);
      }
      //setiap nilai ASCII ditambah 30 pada  pesan  untuk  merepresentasikan karakter
      for ($u=0; $u<strlen($deencrypt); $u+=2) {
          $resultd     .=     chr(substr ($deencrypt,  $u,  2) + 30);
      }
      return $resultd;
  }
```

```
  /*
  * Referensi dari RSA1.3:
  *   http://www.ge.kochi-ct.ac.jp/cgi-
bin-takagi/calcmodp
  *   Flash5   RSA   .fla   by   R.Vijay
<rveejay0 <at> hotmail <dot> com> at
  *
http://www.digitalillusion.co.in/lab/r
saencyp.htm
  */
  ?>
```

Terdapat dua kalkulasi penting untuk kedua fungsi diatas, yaitu

$$X = M^E \pmod N \qquad \textit{func.1}$$

merupakan fungsi untuk melakukan enkripsi terhadap tanda tangan digital. Dan

$$X = M^D \pmod N \qquad \textit{func.2}$$

merupakan fungsi untuk melakukan dekripsi terhadap tanda tangan digital.

III.3 Implementasi SHA dan RSA

Berikut ini merupakan fungsi yang menggunakan gabungan dari implementasi SHA dan RSA, fungsi ini digunakan pada saat pengguna melakukan pendaftaran ke situs :

```
  function    GenerateSHARSA($username,
$password, $salt = null)
  {
      if ($salt === null)
      {
          $salt                          =
substr(md5(uniqid(rand(),   true)),   0,
SALT_LENGTH);
      }
      else
      {
          $salt   =   substr($salt,   0,
SALT_LENGTH);
      }

      $passwordHash                      =
$salt.sha1($salt, $password);
      $keys = generate_keys ();
      //dilakukan   untuk   menambahkan
character ascii
      for           ($i=32;$i<127;$i++)
$passwordHash.=chr($i);
      $encrypt      =      rsa_encrypt
($passwordHash,  $keys[1],  $keys[0]);

      /*kode   ini   bisa   disertakan
disini   untuk   langsung   melakukan
penyimpanan ke basis data*/

      $sql   =   'INSERT   INTO   user
(username,passwordSHARSA,salt,$pub_key
,$priv_key) VALUES (?,?,?,?,?)';
      $result    =    $db->query($sql,
array($username,
$encrypt,$salt,$key[1],$key[1],
$key[2]));

  }
```

Untuk proses verifikasi pada saat *login*, akan digunakan fungsi dibawah ini :

```
  <?php

  /* Nilai Salt telah diketahui */

  $salt = "89e495e";
  $passwordHash                          =
sha1($salt,$_POST['password']);
  $passwordHashReal                      =
$salt.$passwordHash;

  $sql  =  'SELECT  *  FROM  user  WHERE
username = ? AND passwordHash = ?';
  $result      =      $db->query($sql,
array($_POST['username'],
$passwordHash));
  if ($result->numRows() < 1)
  {
      /* Tidak berhasil login */
      echo 'Maaf, informasi yang anda
masukkan salah!';
  }
  else
  {
      while($row                         =
mysql_fetch_assoc($result))
      {
            $priv_key                    =
$row['priv_key'];
            $encrypt                     =
$row['passwordSHARSA'];
            $n = $row['n'];
      }
      for           ($i=32;$i<127;$i++)
$passwordHashReal.=chr($i);
      $decrypt                           =
rsa_decrypt($encrypt,$n,$priv_key);
      if($decrypt                        ==
$passwordHashReal)
      {
            /* Login Berhasil */
            printf('Selamat      datang
%s!', $_POST['username']);
      }
      else
      {
            /* Tidak berhasil login */
            echo 'Maaf, informasi yang
anda masukkan salah!';
      }


  }
```

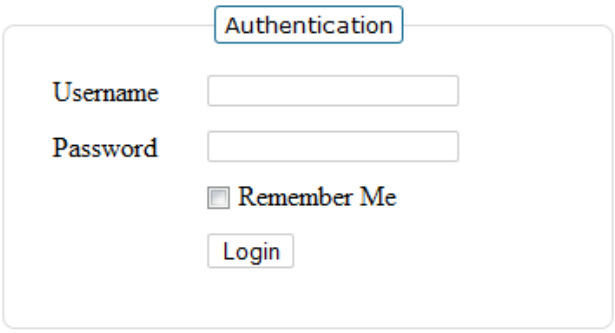
?>

IV. PENGUJIAN

Pengujian dilakukan dengan menggunakan :

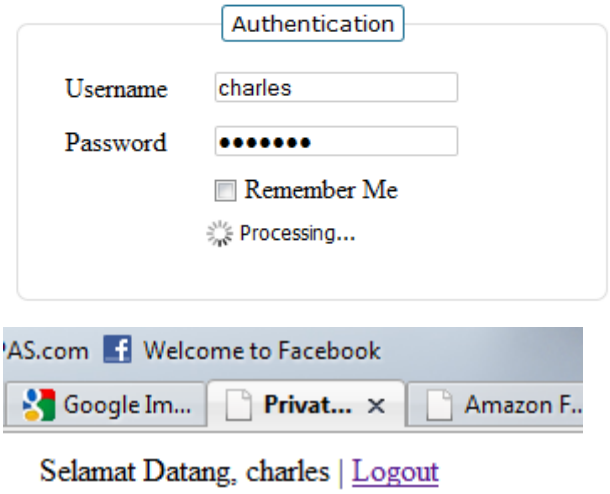
- *XAMPP web server :* dengan PHP 5 dan mysql
- *Client :* bersistem operasi Windows Vista Ultimate, menggunakan Mozilla Firefox 3.6.3
- *Editor :* Notepad++ v.5.1.4
- *Rsa tools :* RSA1.3.php

Berikut ini akan ditampilkan hasil pengujian dengan menggunakan fungsi-fungsi yang telah dibangun.

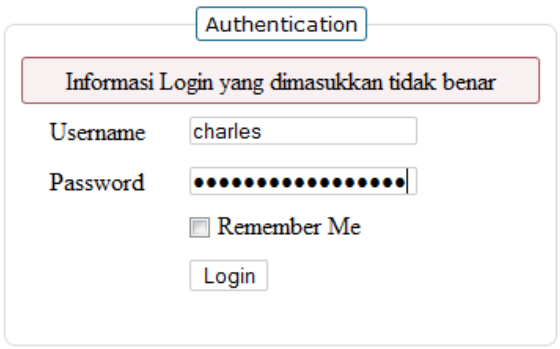


Gambar 5Antarmuka Untuk Memasukkan Informasi Login

Setelah itu pada saat kita memasukkan informasi yang benar, maka akan didapat hasil sebagai berikut :



Gambar 6 Jika Login Berhasil dan Informasi yang Dimasukkan Benar



Gambar 7 Jika Informasi yang Dimasukkan Salah

Informasi kunci dan hasil dekripsi jika login **BENAR**:

Kunci:
P = 5059
Q = 6389
N = 32321951 - modulo
PI = 32310504
E = 7829 - public key
D = 12719501 - private key

```
Test ASCII(32) - ASCII(126):
Password Hash:  !"#$%&'()*+,-
./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRS
TUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxy
z{|}~
Encrypt: 17038396 10878903 3399054
15974036 19215704 27259261 20757885
11600663 12140725 8306515 10012081
8558816 6435041 11311783 4922868
12552065 408662 16591085 24024015
21872953 13308718 13658854 17546006
24478967 17079353 5930329 18481596
11105593 18605073 18373956 17154520
22017406
Decrypt:  !"#$%&'()*+,-
./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRS
TUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxy
z{|}~
Success: True
```

V. KESIMPULAN

Pada penerapan sandi lewat menggunakan SHA + *Salt*, kemungkinan penyerangan terhadap sandi lewat lebih kecil. Penyerang akan membutuhkan waktu yang lama untuk mengetahui kata kunci sebenarnyadari sandi lewat. Hal ini akan memberikan pengamanan ganda pada sandi lewat. Implementasi penggunaan sandi lewat berbasiskan Hash sudah disetujui penggunaannya oleh kalangan industri sebagai salah satu penjaga keamanan sandi lewat. Windows merupakan salah satu pengguna pengamanan menggunakan teknik ini untuk login kedalam Desktopnya. Karena itu penerapan teknik ini

pada hubungan *client server* merupakan sebuah pilihan yang tepat.

Penerapan tanda tangan digital dipastikan menambah integritas dari data dan perubahan-perubahan yang dilakukan pada basis data akan dapat terdeteksi, karena akan terjadi kesalahan selama melakukan login. Kesalahan-kesalahan yang mungkin muncul pada saat validasi menggunakan tanda tangan digital antara lain:

1. Jika terjadi perubahan pada tanda tangan digital, maka akan terdapat pesan kesalahan.
2. Jika terjadi perubahan pada informasi sandi lewat pad saat seorang pengguna telah melakukan login (penyerangan terhadap cookies).
3. Jika terjadi ketidakcocokan antara kunci publik dan kunci private yang ada.

Penerapan tanda tangan digital juga dapat memberikan hasil yang lebih baik, jika diterapkan bersama SHA akan memberikan pengamanan jaringan yang lebih terpercaya. Dan dari hasil percobaan, untuk melakukan login, tidak diperlukan waktu yang lama untuk melakukan kalkulasi selama pendaftaran ataupun pengecekan login.

Dengan pengamanan yang berlapis ini, diharapkan informasi sandi lewat yang merupakan informasi penting setiap orang dapat "Aman" dan "Terjaga". Walaupun pengamanan ini tidak akan luput dari serangan, setidaknya pengamanan ini dapat memperlambat proses pencurian terhadap informasi sandi lewat, sampai ditemukan sebuah pengamanan yang mendekati sempurna.

DAFTAR PUSTAKA

[1] Stallings, William *Cryptography and Network Security Principles and Practices, Fourth Edition.* Prentice Hall 2005.

[2] Azman Samsudin , Chai Wen Chuah, *Omega Network Hash Function.* Journal of Computer Science 2009.

[3] Ragget Malcomm, *Digital Signature and Encryption*. School of Oriental and African Study 2008.

[4] Nakov Stevlin. *How Digital Signatures Works: Digitally Signing Message*. 2009.

[5] McGlinn James, "*Password Hashing*," *PHP Security Consortium 2005*.

[6] Haecker Glenn, Semenov Segey, Suivan, "RSA1.3.php,"*Hacker Hunter Authorization System*.

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, 17 May 2010

ttd

Charles Hariyadi (13505105)