Implementasi Secure Remote Password Protocol (SRP) dengan Modifikasi Penerapan Algoritma Hash SHA-256

Ahmad Zufri -13504044
1) Sekolah Teknik Elektro dan Informatika ITB, Bandung,40132, email: if14044@students.if.itb.ac.id

Abstract *--Semakin berkembangnya teknologi internet sekarang inii membuat banyak masyarakat yang memanfaatkan teknologi internet untuk berbagai keperluan dalam kehidupan sehari – hari. Teknologi internet menyediakan berbagai layanan yang sangat penting sekali seperti layanan internet banking, email, blog, file transfer protocol(FTP) dan juga WWW. Sebagian besar layanan ini membutuhkan proses autentikasi yang dapat menjamin kerahasiaan dari pesan – pesan yang digunakan dalam proses autentikasi terutama password agar tidak terjadi penyalahgunaan oleh orang yang tidak berhak dan ingin mengambil keuntungan untuk dirinya sendiri. Salah satu metode yang digunakan untuk proses autentikasi ialah dengan menggunakan suatu metode yang disebut Secure Remote Password Protocol (SRP) yang memungkinkan seorang user untuk melakukan autentikasi pada suatu server atau penyedia layanan di internet. Secure Remote Password Protocol (SRP) tergolong kedalam algoritma kunci public dan menggunakan fungsi hash untuk membangkitkan kunci session. Dalam makalah ini akan dibahas mengenai implementasi Secure Remote Password Protocol dengan menggunakan algoritma hash SHA-256 yang lebih aman untuk menggantikan algoritma hash SHA-1 sebagai algoritma hash yang digunakan pada SRP standar beserta analisis keamanannya.*

Kata Kunci*:Secure Remote Password Protocol, Hash,autentikasi,internet*

1. PENDAHULUAN

Perkembangan teknologi informasi yang begitu pesat pada saat sekarang ini terutama internet membuat setiap orang menjadi lebih banyak menggunakan berbagai layanan yang disediakan di internet seperti email, internet banking, WWW dan FTP. Hal ini dikarenakan kemudahan dalam menggunakan berbagai layanan tersebut dan hamper semua kebutuhan manusia tersedia di internet. Pada umumnya layanan yang disediakan internet memerlukan suatu proses autentikasi terhadap pengguna yang ingin menggunakan layanan tersebut. Proses autentikasi yang dilakukan harus dapat menjamin kerahasiaan pesan atau parameter yang digunakan dalam proses autentikasi tersebut. Sehingga hanya orang yang berhak untuk melakukan layanan tersebut yang dapat menggunakan layanan yang bersangkutan.

Salah satu cara untuk mengatasi permasalahan diatas ialah dengan menggunakan kriptografi kunci publik untuk melakukan enkripsi terhadap pesan atau parameter yang digunakan yang dikirimkan pada proses autentikasi. Sehingga hanya orang atau pihak yang berhak yang dapat melakukan proses autentikasi.

Salah satu metode yang digunakan untuk proses autentikasi ialah dengan Secure Remote Password Protocol (SRP). SRP menjamin kerahasiaan pesan yang digunakan untuk proses autentikasi. Selain itu SRP juga sangat aman terhadap *Dictionary Attack*. Algoritma ini banyak dipakai dalam proses autentikasi seperti pada layanan ftp, telnet, ssh dan lain – lain. Namun algoritma hash SHA-1 yang digunakan dalam SRP standar seperti yang tercantum dalam RFC 2945 memiliki kelemahan karena memungkinkan terjadinya *Collision* dimana untuk 2 input yang berbeda akan menghasilkan nilai hash yang sama.

Karena hal tersebut penulis berusaha untuk melakukan modifikasi terhadap SRP dengan mengubah algoritna hash yang digunakan dengan algoritma hash SHA-256 yang terbukti lebih aman dan memiliki jumlah bit yang lebih banyak dari algoritma SHA-1.

2.Konsep Dasar

2.1 Secure Remote Password Protocol

Secure Remote Password Protocol (SRP) pertama kali mucul pada tahun 1996 pada USENET. Kemudian mengalami perbaikan pada tahun 1997 dalam hal keamanan. Pada tahun 1998, diperkenalkan SRP-3 yang merupakan hasil pengembangan terhadap SRP sebelumnya setelah melalui diskusi dan perbaikan di bidang yang menyangkut kriptografi. SRP-3 ini kemudian banyak dipakai di berbagai aplikasi dan layanan internet di seluruh dunia.

SRP ialah sebuah protokol yang menangani autentikasi berbasis password. SRP menjamin bahwa autentikasi yang dilakukan oleh computer client ke computer sever aman dengan menggunakan password atau kata kunci rahasia. User dari computer client harus mengingat suatu kunci rahasia yang biasanya disebut password dan

informasi lain yang tidak rahasia. Pada sebagian besar kasus, informasi yang tidak rahasia ini disebut user ID, login ID dan lain – lain. Sedangkan pada sisi server, terdapat sebuah *verifier* untuk setiap user pada server tersebut, yang memungkinkan seorang user untuk melakukan autentikasi pada server tersebut dengan aman. Selain itu, SRP melakukan pertukaran berbagai informasi rahasia yang telah dienkripsi dengan algoritma kriptografi kunci public dalam beberapa tahap tertentu. Jika terjadi kegagalan pada suatu tahap, maka proses autentikasi tidak akan dapat dilanjutkan ke tahap selanjutnya. Hal ini menyebabkan kedua pihak client dan server dapat berkomunikasi dengan aman.

SRP terbukti sangat aman terhadap *dictionary attack*. Yaitu serangan yang ditujukan untuk mendapatkan password dengan cara mencoba semua kata yang ada dalam suatu kamus. Hal ini dilakukan karena biasanya orang memilih kata yang mudah untuk diingat sebagai password. Selain itu, SRP tidak membutuhkan peran dari pihak ke-3 dalam melakukan proses autentikasi. Yang menjadi keunggulan dari SRP ini ialah ketika salah satu aspek kriptografi dalam protokol ini mengalami serangan, SRP masih memiliki keamanan yang bagus. Ini artinya SRP memiliki tingkat keamanan yang berlapis – lapis.

SRP merupakan suatu protocol yang bersifat *open source*. Artinya semua orang dapat menggunakannya tanpa harus mengeluarkan biaya sama sekali.

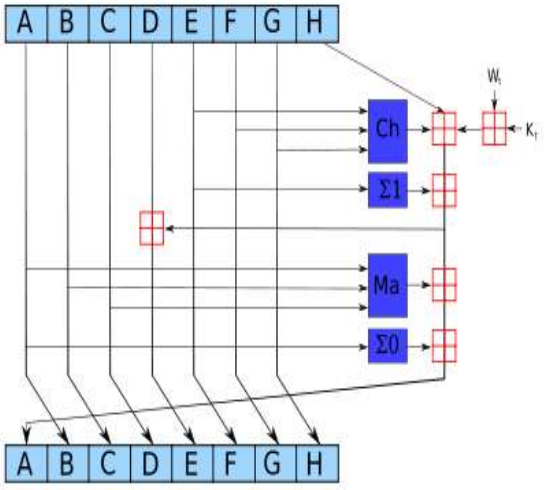
2.2 SHA-256

Algoritma Hash SHA merupakan salah satu dari 5 algoritma hash dalam kriptografi yang diciptakan oleh National Security Agency (NSA) dan dipublikasikan oleh NIST sebagai Amerika Serikat *Federal Information Processing Standard*. SHA merupakan singkatan dari *Secure Hash Algorithm*. Algoritma hash melakukan proses terhadap sebuah masukan dengan panjang berapapun dan menghasilkan sebuah keluaran dengan panjan yang tetap tanpa terpengaruh dari panjang masukan yang diterima. Keluaran dalam fungsi hash serting disebut *message digest*. Perubahan yang sangat kecil pada pesan masukan akan menghasilkan message digest yang berbeda. Suatu algoritma hash dikatakan aman jika:

1. Untuk menemukan pesan masukan dari suatu message digest membutuhkan waktu yang sangat lama dari segi komputasi.
2. Untuk menemukan 2 pesan masukan yang berbeda yang menghasilkan message digest yang sama membutuhkan waktu yang sangat lama dari segi komputasi.

Algoritma hash SHA memiliki bebarapa varian, diantaranya ialah *SHA-1*, *SHA-224*, *SHA-256*, *SHA-384*, dan *SHA-512*. 4 varian terakhir sering juga disebut sebagai SHA-2. SHA-1 menghasilkan message digest dengan panjang 160 bit. Sedangkan pada 4 algortima lain, panjang message digest yang dihasilkan sesuai dengan angka yang terdapat pada namanya.

SHA-256 menghasilkan message digest dengan panjang 256 bit. Hal ini menyebabkan algoritma hash SHA-256 lebih aman dari pada SHA-1. Sampai saat ini belum ada berita mengenai kriptanalisis terhadap SHA-256. Berikut ini skema round yang dipakai pada SHA-256.



SHA-256 terdiri atas 64 kali putaran dan menggunakan 8 buah 32 bit register berbeda dengan SHA-1 yang menggunakan 5 buah 32 bit register dan terdiri atas 80 kali puaran. Pemrosesan pesan dilakukan dalam blok yang panjangnya 512 bit. Jadi pesan yang bertugas sebagai masukan dibagi ke dalam blok – blok sepanjang 512 bit. Selain itu operasi yang dilakukan pada SHA-256 ini lebih banyak dan lebih kompleks dibandingkan dengan operasi yang dilakukan di SHA-1. Penambahan konstanta juga dilakukan dalan SHA-256.

3. Mekanisme SRP dengan modifikasi penerapan algoritma SHA-256

Mekanisme SRP disini pada dasarnya sama dengan mekanisme SRP yang terdapat pada RFC2945. Yang menjadi perbedaan utama disini ialah pada fungsi hash yang digunakan untuk pembangkitan kunci sesi. Pada RFC2945, fungsi hash yang digunakan ialah SHA-1. sedangkan disini fungsi hash yang digunakan ialah SHA-256.

Lambang dan symbol yang digunakan:

N	Bilangan Prima yang sangat besar.
g	generator modulo N
k	parameter perkalian
s	salt
I	Username
p	Password asli sebelum di enkripsi
H()	fungsi hash
^	perpangkatan
u	parameter random
a,b	private key komunikasin
A,B	public key komunikasi
x	Private key (dihitung dari s dan p)
v	Password verifier
K	Kunci Sesi

Mekanisme SRP ialah:

- Server menyimpan password dari user dalam bentuk {<username>,<verifier>,salt}.
- Verifier dihitung dengan rumus
  x = SHA(<salt> | SHA(<username> | ":" | <raw password>))
  <verifier> = v = g^x % N
- Proses komunikasi selanjutnya akan berlangsung sebagai berikut:
  - User -> Host: I, A = g^a (identifikasi user,user mengirimkan A dan username I, a = random number)
  - Host -> User: s, B = kv + g^b (mengirimkan salt s dan B, b = random number)
  - Keduanya akan menghitung: u = H(A, B)
  - User: x = H(s, p) (user enters password)
  - User: S = (B - kg^x) ^ (a + ux) (hitung kunci sesi)
  - User: K = H(S)
  - Host: S = (Av^u) ^ b (hitung kunci sesi)
  - Host: K = H(S)

Selanjutnya Host dan User sudah memiliki kunci sesi yang sama. Untuk melanjtukan proses autentikasi maka Host dan User harus melakukan pemeriksaaan untuk menentukan apakah kunci sesi yang dimiliki oleh Host dan User cocok. Untuk melakukan pemeriksaan maka akan dilakukan proses sebagai berikut:

- User -> Host: $M_1 = H(H(N)$ XOR $H(g)$ | $H(I)$ | s | A | B | K_{user}) Host mencocokkan M1
- Host -> User: $M_2 = H(A$ | M | K_{Host}). User mencocokkan M_2.

Ada beberapa kondisi yang diterapkan dalam protocol ini untuk menjamin keamanan proses autentikasi, yaitu:

1. User akan menghentikan proses jika nilai B yang diterima sama dengan 0 atau nilai u=0.
2. Host akan menghentikan proses jika nilai A yang diterima sama dengan 0
3. User harus mengirimkan pembuktian nilai K nya terlebih dahulu. Jika K yang dikirm User tidak cocok, maka Host harus menghentikan proses tanpa mengirimkan nilai K nya.

Implementasi

Protokol ini telah diimplementasikan oleh penulis dalam bahasa C# dan digunakan untuk proses autentikasi dalam web service.
Berikut ini ialah fungsi dan nilai – nilai parameter yang digunakan dalam implementasi:

- Nilai N yang digunakan

```
public static readonly BigInteger N =new BigInteger(Convert.FromBase64String(prime1024Bit));"7q8Kua2zjdacM/gK+o/F6GByYYd1/zwLnqIxTJwlZXbWdN90luqB0zg7SBPWksbg4NXY4lC5i+SOSVwdYIna0V3H17RhVNa2zo70rWmxXUmCVZspe88YhcUp9WZmDlfsaO28PAVybMAv1Mv0l26qmv1ROP6DdkNbn8YdL8DrBuM="
```

- Nilai g yang digunakan

```
public static readonly BigInteger g1024Bit = new BigInteger(2);
```

Kode Program di sisi Client ialah:

```
private SrpReply GetSctSrp(string
userName, string password, out
byte[] key)
        {
                        SoapEnvelope
se = new SoapEnvelope();

            // SRP Client Process
            BigInteger a =
SRPFunctions.Geta();
            BigInteger A =
SRPFunctions.CalcA(N, g, a);
            SrpRequest request =
new SrpRequest();
            request.UserName =
userName;
            request.A =
A.GetBytes();

se.SetBodyObject(request);
SrpReply reply =
(SrpReply)base.SendRequestResponse
("GetSctSrp",
se).GetBodyObject(typeof(SrpReply)
);
BigInteger B = new
BigInteger(reply.B);
            if ( (B % N) == 0 )
                throw new
Exception("B mod N is zero.");

            // Both sides calc u,
S and K.
            BigInteger k =
SRPFunctions.Calck(N, g);
            BigInteger u =
SRPFunctions.Calcu(A, B,
N.ToString().Length);
            if ( u == 0 )
                throw new
Exception("u is zero.");
            BigInteger x =
SRPFunctions.Calcx(reply.Salt,
userName, password);
            BigInteger S =
SRPFunctions.CalcSClient(N, g, B,
k, x, a, u);
            byte[] K =
SRPFunctions.CalcK(S);
            byte[] M =
SRPFunctions.CalcM(N, g, userName,
reply.Salt, A, B, K);
            se = new
SoapEnvelope();
            VerifyRequest vr = new
VerifyRequest();
            vr.SctIdentifier =
reply.SCTIdentifier;
            vr.M = M;
            se.SetBodyObject(vr);
byte[] serversM2 =
(byte[])base.SendRequestResponse("
VerifyKey",
se).GetBodyObject(typeof(byte[]));
            byte[] clientsM2 =
SRPFunctions.CalcM2(A, M, K);
            if ( !
Utils.ArraysEqual(serversM2,
clientsM2) )
                throw new
Exception("Invalid login attempt.
Username or password invalid.");

Console.WriteLine("Server proof is
good.");
            string keyString =
BitConverter.ToString(K);
            key = K;
            return reply;
```

Kode Program di sisi server ialah:

```
public static SrpReply
GetSctSrp(SrpRequest request)
        {
            if ( request == null )
                throw new
ArgumentNullException("request");
            if ( logonManager ==
null )
                throw new
InvalidOperationException("LogonMa
nager must be set before calling
this method.");
            if ( request.UserName
== null )
                throw new
ArgumentNullException("request.Use
rName");
            if ( request.A == null
)
                throw new
ArgumentNullException("request.A")
;

            SrpReply sr = new
SrpReply();
            BigInteger A = new
BigInteger(request.A);

            if ( (A % N) == 0 )
                throw new
Exception("A mod N is zero.");


            string saltString =
null;
            string vString = null;
            try
            {

logonManager.LookupUser(request.Us
```

```
erName, out vString, out
saltString);
                if ( saltString ==
null || vString == null )
                    throw new
Exception("LogonManager.LookupUser
method returned a null verifier or
null salt value.");
            }
            catch ( Exception ex )
            {

logonManager.OnLogonUserFailed(req
uest.UserName, ex);
                throw ex;
            }
            BigInteger v = new
BigInteger(Convert.FromBase64Strin
g(vString));
            byte[] salt =
Convert.FromBase64String(saltStrin
g);
            BigInteger b =
SRPFunctions.Getb();
// Get new private b (random
secret ephemeral value).
            BigInteger B =
SRPFunctions.CalcB(N, g, b, v);
// Server calcs public B using
args.
            sr.Salt = salt;
// Return salt to the user.
            sr.B = B.GetBytes();
// Return public B to the user.

            // Both sides
calculate u, S and K.
            BigInteger u =
SRPFunctions.Calcu(A, B,
N.ToString().Length);
            BigInteger S =
SRPFunctions.CalcSServer(N, A, v,
u, b);
            byte[] K =
SRPFunctions.CalcK(S);
            string keyString =
BitConverter.ToString(K);

UsernameToken ut = new
UsernameToken(request.UserName,
"1");
            GenericIdentity gi =
new
GenericIdentity(request.UserName);
            string[] roles = new
string[] { "user", "admin" };
            GenericPrincipal gp =
new GenericPrincipal(gi, roles);
ut.Principal = gp;
            SecurityContextToken
sct = new
SecurityContextToken(ut);
            sct.KeyBytes =
Utils.GetBytes(K, 16);
            DateTime expires =
DateTime.Now.AddMinutes(SctExpireM
inutes);
            sct.LifeTime = new
LifeTime(DateTime.Now, expires);

            lock(syncRoot)
            {
                SRPInfo srpInfo =
new SRPInfo();
                srpInfo.A = A;
                srpInfo.B = B;
                srpInfo.UserName =
request.UserName;
                srpInfo.Salt =
salt;
                srpInfo.K = K;
                srpInfo.SCT = sct;
                sr.SCTIdentifier =
sct.Identifier;
                sr.SCTExpires =
Utils.ToUtcDateTimeString(expires.
ToUniversalTime());

srpList[sct.Identifier] = srpInfo;
            }
            return sr;
        }
```

Algoritma Hash SHA-256 ialah:

```
private void processMessage()
        {
            int num_blok =
message.Length / 64;
            uint[] h = new
uint[4];
            byte[] t = new
byte[4];
            byte[] chunk = new
byte[64];
            uint[] blok_chunk =
new uint[16];
            uint f = 0, g = 0, tmp
= 0;
            for (int i = 0; i <
num_blok; i++)
            {
               for (int
j=16;j<=63;j++)
                    {
                       s0 =
(rightrotate(w[j-15] , 7)) ^ (
rightrotate(w[j-15], 18)) ^ (
rightshift (w[j-15],3))
                       s1 =
(rightrotate(w[j-2], 17)) ^
(rightrotate (w[j-2],19)) ^
```

```
(rightshift(w[j-2] ,10))
                          w[j] =
w[j-16] + s0 + w[j-7] + s1
                        }
                  a = h0
                  b = h1
                  c = h2
                  d = h3
                  e = h4
                  f = h5
                  g = h6
                  h = h7
                  for (int
k=0;k<=63;k++)
                    {
                      s0 =
(rightrotate(a, 2)) ^
(rightrotate(a, 13)) ^ (
rightrotate(a, 22))
                      maj = (a and
b) ^ (a and c) ^ (b and c)
                      t2 = s0 + maj
                      s1 =
(rightrotate(e, 6)) ^
(rightrotate(e, 11)) ^
(rightrotate (e,25))
                      ch = (e and f)
^ ((!e) and g)
                      t1 = h + s1 +
ch + k[i] + w[i]
                      h = g
                      g = f
                      f = e
                      e = d + t1
                      d = c
                      c = b
                      b = a
                      a = t1 + t2
                        h0 = h0 + a
                        h1 = h1 + b
                        h2 = h2 + c
                        h3 = h3 + d
                        h4 = h4 + e
                        h5 = h5 + f
                        h6 = h6 + g
                        h7 = h7 + h
                    }
                    P[8]=new int[8];
P[0]=h0;P[1]=h1;P[2]=h2;P[3]=h3;P[
4]=h4;P[5]=h5;P[6]=h6;P[7]=h7;
hash = Concat(P);
            }
        }
```

Kesimpulan Implementasi SRP

SRP merupaka protocol yang banyak digunakan di berbagai aplikasi internet karena mudah untuk diimplementasikan. SRP merupakan sistem autentikasi berbasis password yang memiliki aspek keamanan yang sangat tinggi karena password tidak di transmisikan melainkan di enkripsi terlebih dahulu sehingga bebas dari man in the middle attack dan juga bebas dari dictionary attack karena jika ada nilai yang tidak sesuai yang dikirimkan oleh client atau server maka SRP akan mengakhiri proses autentikasi dan harus diulang dari awal lagi.

Algoritma SHA-256 dapat dengan mudah dipakai pada SRP tanpa mengubah struktur dan mekanisme protocol. Sehingga kita dapat menggunakan fungsi hash yang paling aman atau sesuai dengan kebutuhan.

Algoritma Hash SHA-256 sampai sekarang ini masih sangat aman karena belum ditemukan adanya kriptanalisis dan collision pada algoritma ini.

Referensi:

[1]Diktat mata kuliah Kriptografi,RInaldi Munir
[2] RFC 2945
[3]http://en.wikipedia.org diakses pada bulan desember 2007
[4]http://guru.multimedia.cx/crc32-vs-adler32/
[5]http://srp.stanford.edu