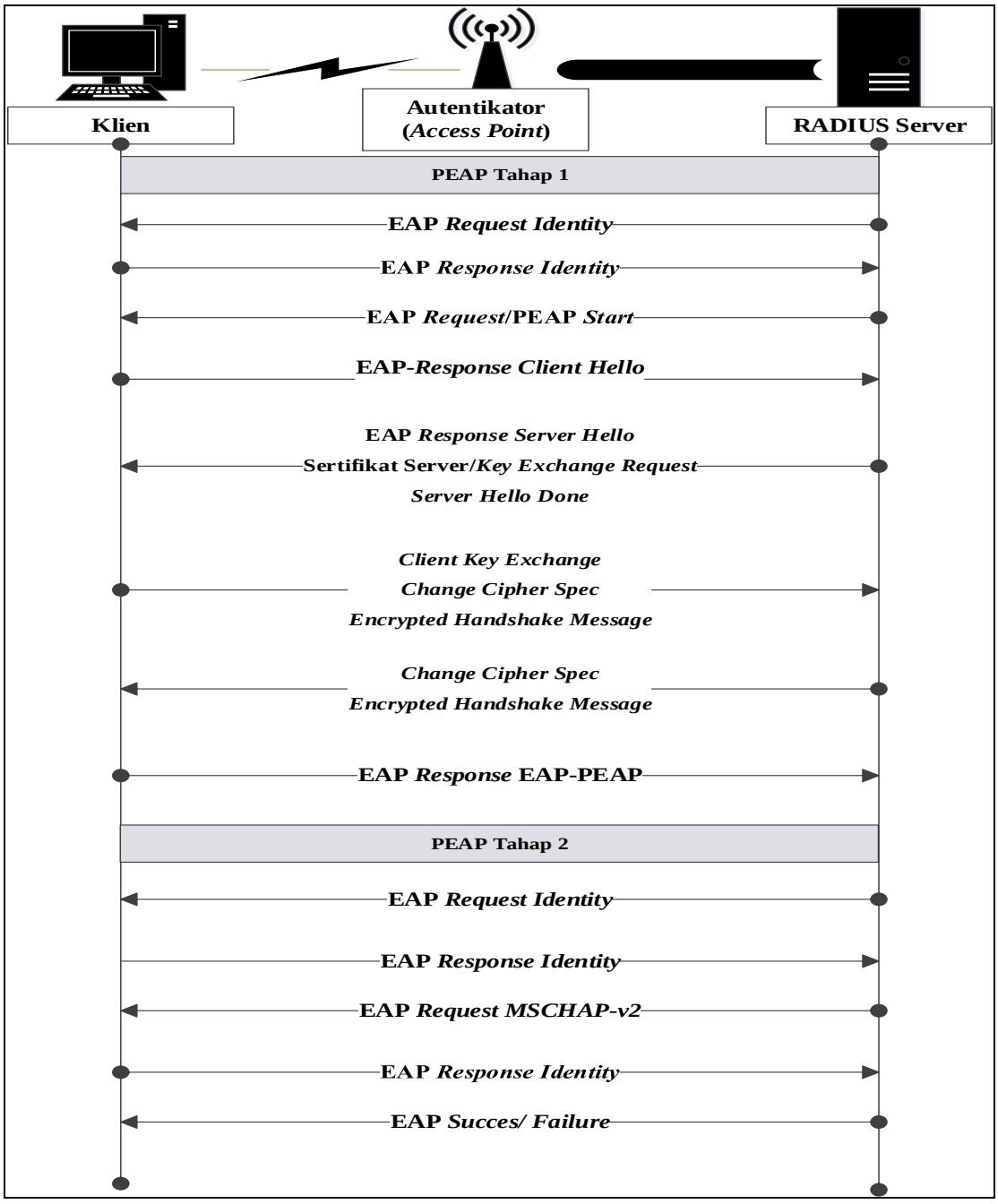


Gambar 2.5. PEAP Tahap 1 dan 2

2.4. Kriptografi Asimetris

Kriptografi asimetris adalah algoritma yang menggunakan kunci yang berbeda untuk proses enkripsi dan deskripsi. Dimana kunci enkripsi dapat disebarkan kepada umum dan dinamakan sebagai kunci publik (*public key*), sedangkan kunci dekripsi disimpan untuk digunakan sendiri dan dinamakan sebagai kunci pribadi (*private key*). Oleh karena itu, kriptografi ini dikenal pula dengan nama kriptografi kunci publik (*public key cryptography*). Adapun contoh algoritma yang menggunakan kunci asimetris adalah RSA (Rivest Shamir Adleman). Pada kriptografi asimetris, dimana setiap pihak akan memiliki sepasang kunci, yaitu kunci publik dan kunci pribadi, dimana kunci publik didistribusikan kepada umum, sedangkan kunci pribadi disimpan untuk diri sendiri. Artinya, bila A ingin mengirimkan pesan kepada B, A dapat menyandikan pesannya dengan menggunakan kunci publik B, dan bila B ingin membaca surat tersebut, ia perlu mendekripsikan surat itu dengan kunci privatnya. Dengan demikian kedua belah pihak dapat menjamin asal surat serta keaslian surat tersebut.

Algoritma RSA merupakan Algoritma yang menggunakan kriptografi asimetris yaitu dengan menggunakan kunci publik dan kunci privat. Algoritma RSA adalah sebuah algoritma yang bekerja per-blok data dimana *plaintext* dan *ciphertext* adalah bilangan bulat antara 0 dan *n*-1 untuk sebuah *n* tertentu. Algoritma ini bekerja dengan menghitung eksponen dari *plaintext* dalam operasi modulo. Sebelum *plaintext* dikirimkan, *plaintext* tersebut harus dienkripsi dengan cara dipangkatkan dengan sebuah bilangan sehingga menjadi sebuah *ciphertext* kemudian *ciphertext* tersebut harus dipangkatkan dengan sebuah bilangan untuk memperoleh *plaintext*. Secara garis besar proses enkripsi oleh algoritma RSA dibagi ke dalam dua proses yaitu, membangkitkan kunci publik dan kunci privat dan melakukan enkripsi dengan kedua kunci tersebut [13]. Algoritma untuk membangkitkan kunci publik dan kunci privat adalah sebagai berikut:

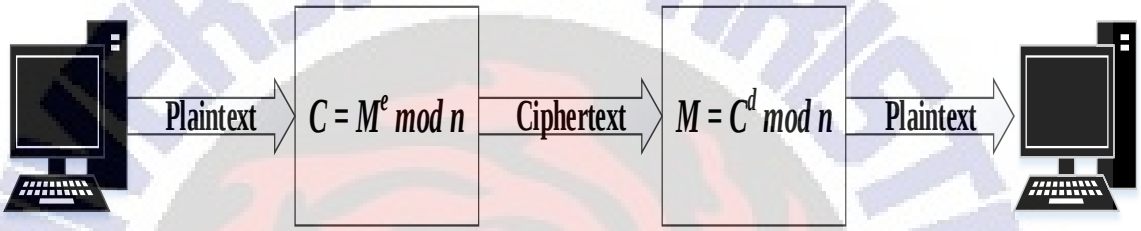
1. Pilih secara acak 2 buah bilangan prima, p dan q.
2. Hitung nilai $n=p.q$. Dimana p dan q sebaiknya tidak merupakan bilangan yang sama.
3. Hitung nilai $\varphi(pq) = (p - 1)(q - 1)$
4. Hitung nilai d dan e dengan rumus $ed \bmod \varphi(pq) = 1$.
5. Buat kunci publik (e,n) yang kemudian disebarkan yang relatif prima dengan $\varphi(pq)$.

6. Buat kunci privat *(d,n)* yang kemudian disimpan dengan menggunakan rumus *ed* mod $\varphi(n) = 1$

Setelah kunci publik dan kunci privat sudah dibuat maka bisa dilakukan proses enkripsi dan dekripsi dengan rumus:

$C = M^e \bmod n$

$M = C^d \bmod n$



Gambar 2.6. Algoritma RSA

2.5. *Secure Socket Layer* (SSL)

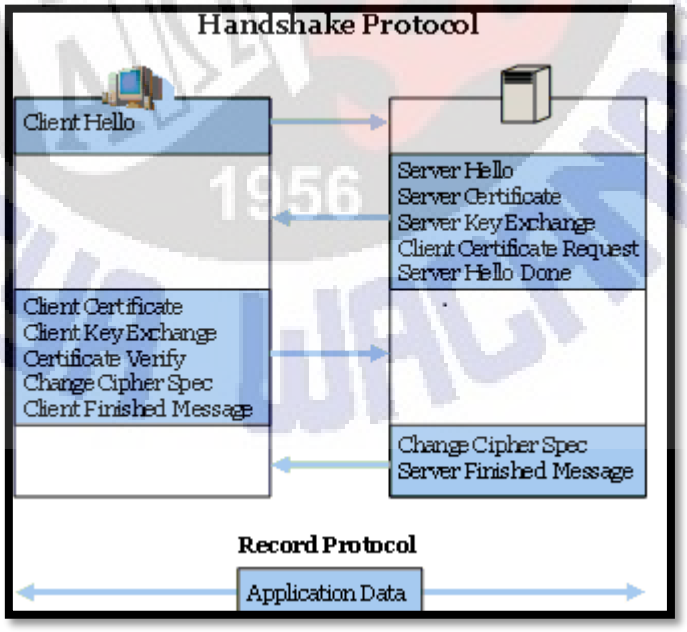
SSL adalah teknologi keamanan standar untuk mendirikan sebuah jembatan antara klien dan server. SSL mengizinkan *Public Key Infrastructure* (PKI) untuk berjalan pada suatu jaringan. *Public Key Infrastructure* (PKI) merupakan serangkaian aturan, kebijakan, dan prosedur untuk membuat, mengatur, mendistribusikan, dan membatalkan sertifikat digital dan mengatur enkripsi kunci publik. Tujuan dari PKI ini adalah untuk mengamankan pengiriman informasi secara elektronik [14].

Ada 4 lapisan dari protokol SSL. Yaitu

1. *Handshake Protokol*
2. *Record Layer Protoko*l pada SSL
3. *Change Cipher Spec Layer*
4. *Alert Protokol* SSL

Ada 2 protokol yang paling penting di dalam SSL adalah *Record Layer Protocol* dan *Handshake Protocol*. *Protocol* SSL *Record Layer* digunakan untuk membungkus data yang dikirimkan di terima setelah protokol *handshake* digunakan untuk membangun parameter keamanan saat terjadi pertukaran data. Tahapan *SSL handshake* ditunjukan pada Gambar 2.6.

1.	Klien mengirim pesan *Client Hello*
2	Server melakukan verifikasi dengan pesan server hello
3	Server mengirim sertifikat
4	Opsional : Server meminta sertifikat klien
5	Opsional : klien memberikan sertifikat klien
6	Klien mengirimkan pesan *Client Key Exchange*
7	Klien mengirmkan pesan *Certificate verify*
8	Pertukaran pesan *Change Cipher Spec*



Gambar 2.7. Protokol SSL *Handshake* [14].

1. *Client-Hello Message*

   Isi pesan *client hello message*

   a. *Ssl Version Number*: klien mengirim *list* dari versi SSL yang didukung. Prioritas diberikan ke versi terbaru yang didukung.
   b. *Random Data Number*: Terdiri dari 32 *Byte*. 4 *Byte* angka dari waktu dan tanggal dari klien dan angka acak.
   c. *Session Id*: untuk mengidentifikasi setiap sesi dalam pertukaran pesan.
   d. *Cipher Suits*: algoritma RSA yang digunakan untuk pertukaran kunci yang akan digunakan untuk kriptografi kunci publik.
   e. *Compression Algorithm*: berisi algoritma kompresi jika digunakan.

2. *Server Hello Message*

   1. *Server Hello*

      Isi Pesan *server hello*:

   a. *Version Number*: server memilih versi dari SSL yang didukung oleh server maupun klien.
   b. *Random Data* :server juga mengeluarkan nilai acak menggunaka 4 *Byte* waktu dan tanggal ditambah angka acak 28 *Byte*.
   c. *Session Id*: ada 3 kemungkinan yang terjadi pada *session Id* tergantung pada pesan *client-hello*. Jika klien memerlukan untuk melanjutkan sesi sebelumnya maka keduanya akan menggunakan *Id* yang sama. Jika klien menghendaki sesi baru maka server membuat sesi baru. Terkadang ada juga sesi *null* jia server tidak melanjutkan sesi.
   d. *Cipher Suits*: hampir sama dengan *version number* dari server, server akan memilih *cipher suits* yang didukung kedua belah pihak.

2. Sertifikat Digital.

Sertifikat digital adalah dokumen yang ditandatangani secara digital yang berisi kunci publik dan informasi penting mengenai jati diri pemilik kunci publik, seperti misalnya nama, alamat, pekerjaan, jabatan, perusahaan dan bahkan *hash* dari suatu informasi rahasia yang ditandatangani oleh suatu pihak terpercaya. Sertifikat digital tersebut ditandatangani oleh sebuah pihak yang dipercaya yaitu *Certificate Authority* (CA). Sertifikat digital atau yang biasa disebut sertifikat kunci publik merupakan bagian dari *Public Key Infrastructure* (PKI). Jenis sertifikat yang umum digunakan adalah X.509 [13].

Tujuan utama dalam pembuatan sertifikat digital adalah untuk memastikan bahwa kunci publik adalah milik dari dari seseorang atau entitas yang ingin kita koneksikan.

Version
Serial Number
Signature Algorithm
Issure Name
Period of Validity
Subject Name
Subject Public Key
Extension
Signature

Gambar 2.8. Format Sertifikat X.509

3. *Server Key Exchange*: tahap ini ditangani oleh server, jika tidak ada kunci publik yang dibagikan bersama dengan sertifikat.
4. *Client Certificate Request*: jarang digunakan karena hanya digunakan ketika klien melakukan autentikasi dengan sertifikat klien.

5. *Server Hello Done*: pesan ini dikirim oleh server ketika server ingin memberitahu klien bahwa server telah selesai mengirim pesan *hello* dan menunggu respon dari klien.

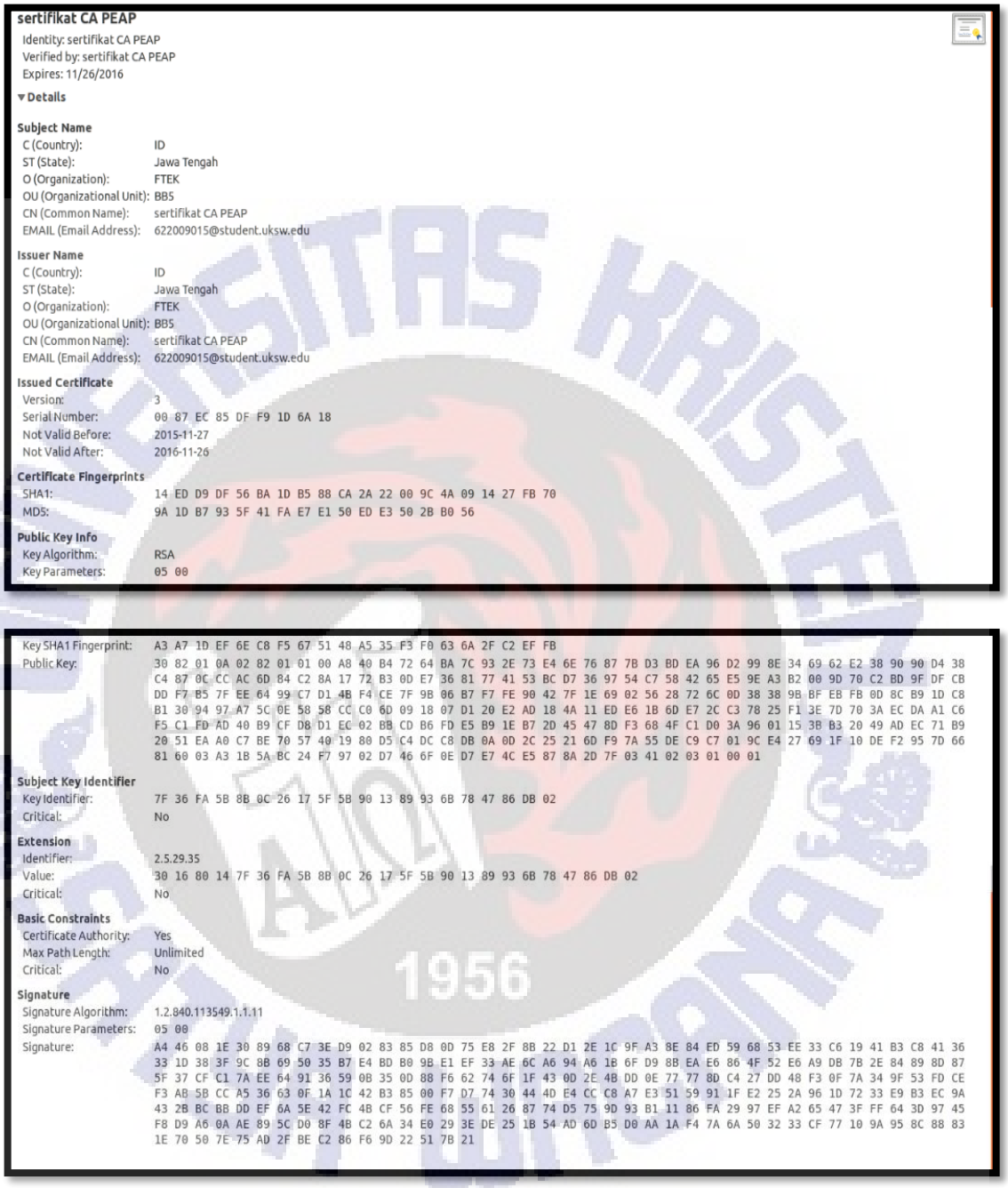
   Respon dari klien pada pesan *Server Hello Message*:

   Client Certificate: klien mengirim sertifikat klien kepada server. Tahap ini dijalankan hanya jika server meminta sertifikat klien.

   Client Key Exchange: pesan dikirim ketika klien selesai menghitung *secret* dengan bantuan angka acak dari server dan klien. Pesan ini dikirm dengan mengenkripsi dengan kunci publik yang dibagikan melalui *hello message*. Pesan ini hanya bisa didekripsi dengan *private key* server.

   Change Cipher Spec: Pesan ini dikirim oleh klien kepada server untuk mengindikasikan bahwa server harus mengirimkan pesan berikutnya dalam format yang sudah terenkripsi. Paket ini merupakan paket terakhir yang bisa dibaca.

Gambar 2.9. Sertifikat Digital pada Sistem Operasi Windows dengan format .der



Gambar 2.10. Sertifikat Digital pada Sistem Operasi Linux Ubuntu dengan Format .pem

Gambar 2.11. Sertifikat Digital yang di-*capture* Menggunakan Perangkat Lunak *Wireshark*