Sicurezza dei Sistemi Informatici

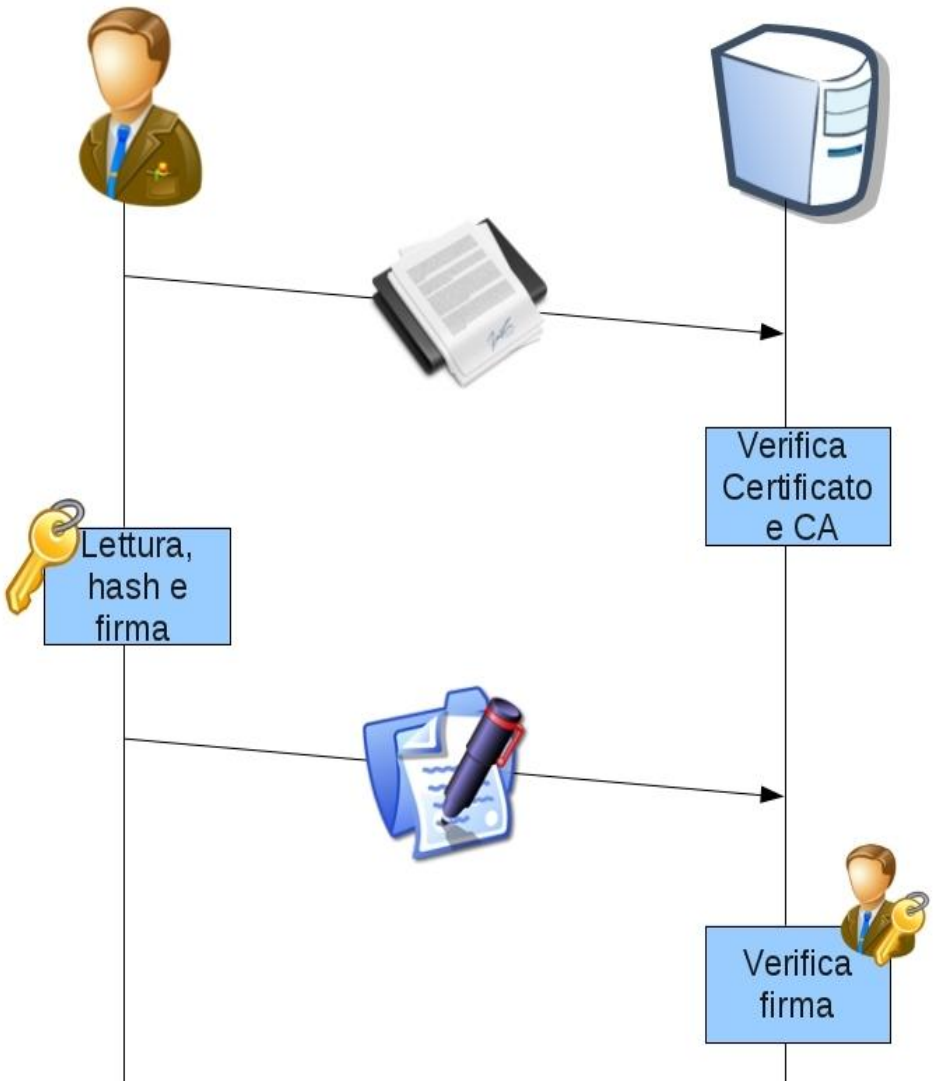
Esercitazioni OpenSSL

Firma digitale con RSA

Roberta Daidone

roberta.daidone@iet.unipi.it

Obiettivo



- Il client invia il proprio certificato al server.
- Il server verifica il certificato del client e possiede il certificato della CA.
- Il client firma l'hash del file con la propria chiave privata.
- Il server riceve il file e verifica la firma con la chiave pubblica del client.

Funzioni client/server

- #include <openssl/evp.h>
  #include <openssl/pem.h>

- Allocazione contesto

```
char* alg="sha1";
const EVP_MD* md = EVP_get_digestbyname(alg);
OpenSSL_add_all_digests();
EVP_MD_CTX * ctx;
ctx = malloc(sizeof(EVP_MD_CTX));
```

- Deallocazione contesto

```
EVP_MD_CTX_cleanup(ctx);
free(ctx);
```

Funzioni client

- Preparazione contesto

```
EVP_MD_CTX_init(ctx);
EVP_SignInit(ctx, md);
```

- Allocazione di memoria per la chiave privata

```
EVP_PKEY* priv_key;
priv_key = (EVP_PKEY*) malloc(sizeof (EVP_PKEY));
```

- Lettura della chiave privata per la firma

```
EVP_PKEY*  PEM_read_PrivateKey(FILE  *fp,  EVP_PKEY  **x,
   pem_password_cb *cb, void *u)
```

fp file contenente la chiave privata
x buffer dove verrà posta la chiave privata letta
cb funzione di callback per il recupero della password
u parametri della funzione di callback (password)

Funzioni client

- Firma
  - EVP_SignUpdate(ctx, buffer, size);

    buffer contiene il file di cui si intende fare hash e firma;
    size specifica di quanti byte di buffer voglio fare l'hashing.

    La funzione di update completa l'hashing in più mandate

  - EVP_SignFinal(ctx,sign_buffer, sign_buffer_size, priv_key);

    sign_buffer torna al chiamante l'hash firmato;
    sign_buffer_size torna al chiamante la dimensione (in byte) della firma digitale.

Funzioni server

```
#include <openssl/x509_vfy.h>
#include <openssl/x509.h>
#include <openssl/pem.h>
#include <openssl/evp.h>
```

- Preparazione contesto

```
EVP_MD_CTX_init(ctx);
EVP_VerifyInit(ctx, md);
```

- Allocazione di memoria per la chiave pubblica

```
EVP_PKEY* pub_key;
pub_key = (EVP_PKEY*) malloc(sizeof (EVP_PKEY));
```

Funzioni server

• Verifica del certificato del client e verifica della CA che lo ha emesso

```
FILE* file;
X509* cert;
X509_STORE* store;
X509_LOOKUP* lookup;
X509_STORE_CTX* verify_ctx;
```

cert struttura dati per il certificato del client;
store gestisce una collezione di certificati (la chain of trust);
lookup permette di fare lookup dei certificati della chain of trust;
verify_ctx è il contesto di verifica della chain of trust fino al certificato dell'utente.

Funzioni server

- Leggere il certificato inviato dal client

  cert = PEM_read_X509 (file, NULL, NULL, NULL);

  2 parametro = certificato da leggere
  3 parametro = funzione di callback per la password
  4 parametro = argomenti della funzione di callback

- Creazione del **cert store**

  store = X509_STORE_new();

- Leggere il certificato della CA e caricare la directory di riferimento della CA nel **cert store**

  X509_STORE_load_locations (store, ca_file, dir);

  dir = "./exampleCA"
  ca_file = "cacert.pem"

Funzioni server

• Specificare che il path di ricerca dei certificati è quello specificato nel **cert store**

X509_STORE_set_default_paths(store);

• In mancanza della chiamata precedente, il path di default è usr/local/ssl/certs.

• Caricare nel cert store il metodo di lookup dei certificati specificato dalla funzione **X509_LOOKUP_file()**

lookup=X509_STORE_add_lookup(store,X509_LOOKUP_file())

• **X509_LOOKUP_file()** torna un puntatore al metodo di default che è *x509_file_lookup* definito dalle openSSL.

Funzioni server

- Caricare la CRL
  X509_load_crl_file(lookup, CRLfile, X509_FILETYPE_PEM);
  CRLfile = "CRLfile.pem"

- Creazione di un verification context
  verify_ctx = X509_STORE_CTX_new();

- Inizializzazione del verification context
  X509_STORE_CTX_init(verify_ctx, store, cert, NULL);

- Verificare il certificato del client
  X509_verify_cert(verify_ctx);

- Deallocazioni
  X509_free(cert);
  X509_STORE_free(store);
  X509_STORE_CTX_free(verify_ctx);

Funzioni server

- Recupero della chiave all'interno del certificato

  *pub_key = X509_get_pubkey(cert);

- Verifica della firma digitale
  - EVP_VerifyUpdate(ctx, buffer, size);

  buffer contiene il file di cui voglio fare hash e verificare la firma

  size specifica di quanti byte di buffer voglio fare l'hashing.

  La funzione di update completa l'hashing per la verifica in più mandate

Funzioni server

- EVP_VerifyFinal(ctx,sign_buffer,sign_buffer_size,pub_key)

  sign_buffer contiene l'hash firmato inviato dal client;
  sign_buffer_size contiene la dimensione (in byte) della firma digitale del client.

  La funzione confronta il contenuto di *sign_buffe*r con l'hashing contenuto in *ctx*, verificato con *pub_key*.
  Ritorna 1 se il confronto ha successo.

Esercizio

- Si consideri la firma digitale con RSA.

- Sfruttare le chiavi e i certificati generati la scorsa lezione.
  - Il client ha il proprio certificato e la propria chiave privata
  - Il server ha il certificato e la CRL della CA
  - Il client invia al server il proprio certificato

- Il client:
  - legge e invia il proprio certificato al server
  - legge un file *F*, e invia al server il suo contenuto seguito dalla firma digitale.

- Il server:
  - riceve e verifica il certificato del client grazie al certificato della CA che lo ha emesso
  - recupera la chiave pubblica del client
  - riceve il file del client, ne verifica la firma e lo salva su file.