OpenSSL Certificates

Michele La Manna
Dept. of Information Engineering
University of Pisa
michele.lamanna@phd.unipi.it
Version: 2021-05-12

Exercise

CERTIFICATES

Some preps before the exercise

UNIVERSITÀ DI PISA

Set up SimpleAuthority CA.

- Export CA's self-signed root certificate in PEM format.
- Create a certificate for a subject (e.g., a server).
- Export subject's certificate in PEM format.
- Export subject's private key in PEM format.
- Export CRL in PEM format.

Certificates exercise

UNIVERSITÀ DI PISA

Write a "sign" program, which impersonates a Server and:

1. Loads the private key associated with the generated certificate;
2. Loads a file (e.g. "file.txt");
3. Signs said file with the previously loaded private key;
4. Writes the signature in a (e.g., "file.txt.sgt").

Write a "certvfy" program, which impersonates the client and:

1. Loads a CA's certificate and a CRL.
2. Builds a store with CA's certificate and CRL.
3. Loads and verifies the server's certificate; outputs the outcome, the subject's distinguished name and the CA's distinguished name.
4. If verification went well, loads "file.txt" and "file.txt.sgt".
5. Verifies the signature over the loaded file.

Desired Output

```
cybersecurity@cybersecurity-VirtualBox:~/Scrivania/FoC/lab10 cert/cert$ g++ sign.cpp -o server.out -lcrypto
cybersecurity@cybersecurity-VirtualBox:~/Scrivania/FoC/lab10 cert/cert$ g++ certvfy_sgntvfy.cpp -o client.out -lcrypto
cybersecurity@cybersecurity-VirtualBox:~/Scrivania/FoC/lab10 cert/cert$ ./server.out
Please, type the PEM file containing my private key: ChatApp_key.pem
Please, type the file to sign: file.txt
File 'file.txt' signed into file 'file.txt.sgn'
cybersecurity@cybersecurity-VirtualBox:~/Scrivania/FoC/lab10 cert/cert$ ./client.out
Please, type the PEM file containing peer's certificate: ChatApp_cert.pem
Certificate of "/C=IT/CN=ChatApp" (released by "/C=IT/O=FoC/OU=Certification Authority/CN=FoundationsOfCybersecurity") verified succe
ssfully
Please, type the signature file: file.txt.sgn
Please, type the file to verify: file.txt
The Signature has been correctly verified! The message is authentic!
cybersecurity@cybersecurity-VirtualBox:~/Scrivania/FoC/lab10 cert/cert$
```

You can also choose to use some hard-coded file names.

In the example, the CA certificate and the CRL are automatically loaded by the client.

Digital Signature

UNIVERSITÀ DI PISA

Write a "sign" program, which impersonates a Server and:
1) Loads the private key associated with the generated certificate;
2) Loads a file (e.g. "file.txt");
CP! 3) Signs said file with the previously loaded private key;
4) Writes the signature in a (e.g., "file.txt.sgt").

15 minutes

Server Checkpoint 3

UNIVERSITÀ DI PISA

```
50    // declare some useful variables:
51    const EVP_MD* md = EVP_sha256();
52
53    // create the signature context:
54    EVP_MD_CTX* md_ctx = EVP_MD_CTX_new();
55    if(!md_ctx){ cerr << "Error: EVP_MD_CTX_new returned NULL\n"; exit(1); }
56
57    // allocate buffer for signature:
58    unsigned char* sgnt_buf = (unsigned char*)malloc(EVP_PKEY_size(prvkey));
59    if(!sgnt_buf) { cerr << "Error: malloc returned NULL (signature too big?)\n"; exit(1); }
60
61    // sign the plaintext:
62    // (perform a single update on the whole plaintext,
63    // assuming that the plaintext is not huge)
64    ret = EVP_SignInit(md_ctx, md);
65    if(ret == 0){ cerr << "Error: EVP_SignInit returned " << ret << "\n"; exit(1); }
66    ret = EVP_SignUpdate(md_ctx, clear_buf, clear_size);
67    if(ret == 0){ cerr << "Error: EVP_SignUpdate returned " << ret << "\n"; exit(1); }
68    unsigned int sgnt_size;
69    ret = EVP_SignFinal(md_ctx, sgnt_buf, &sgnt_size, prvkey);
70    if(ret == 0){ cerr << "Error: EVP_SignFinal returned " << ret << "\n"; exit(1); }
71
72    // delete the digest and the private key from memory:
73    EVP_MD_CTX_free(md_ctx);
74    EVP_PKEY_free(prvkey);
```

Digital Signature

UNIVERSITÀ DI PISA

Write a "certvfy" program, which impersonates the client and:
CP! 1) Loads a CA's certificate and a CRL.

10 minutes

Client Checkpoint 1

UNIVERSITÀ DI PISA

```
// load the CA's certificate:
string cacert_file_name="FoundationsOfCybersecurity_cert.pem";
FILE* cacert_file = fopen(cacert_file_name.c_str(), "r");
if(!cacert_file){ cerr << "Error: cannot open file '" << cacert_file_name << "' (missing?)\n"; exit(1); }
X509* cacert = PEM_read_X509(cacert_file, NULL, NULL, NULL);
fclose(cacert_file);
if(!cacert){ cerr << "Error: PEM_read_X509 returned NULL\n"; exit(1); }

// load the CRL:
string crl_file_name="FoundationsOfCybersecurity_crl.pem";
FILE* crl_file = fopen(crl_file_name.c_str(), "r");
if(!crl_file){ cerr << "Error: cannot open file '" << crl_file_name << "' (missing?)\n"; exit(1); }
X509_CRL* crl = PEM_read_X509_CRL(crl_file, NULL, NULL, NULL);
fclose(crl_file);
if(!crl){ cerr << "Error: PEM_read_X509_CRL returned NULL\n"; exit(1); }
```

Digital Signature

UNIVERSITÀ DI PISA

Write a "certvfy" program, which impersonates the client and:
CP! 2) Builds a store with CA's certificate and CRL.

10 minutes

Client Checkpoint 2

UNIVERSITÀ DI PISA

```
33  // build a store with the CA's certificate and the CRL:
34  X509_STORE* store = X509_STORE_new();
35  if(!store)
36  { cerr << "Error: X509_STORE_new returned NULL\n" << ERR_error_string(ERR_get_error(), NULL) << "\n"; exit(1); }
37  ret = X509_STORE_add_cert(store, cacert);
38  if(ret != 1)
39  { cerr << "Error: X509_STORE_add_cert returned " << ret << "\n" << ERR_error_string(ERR_get_error(), NULL) << "\n"; exit(1); }
40  ret = X509_STORE_add_crl(store, crl);
41  if(ret != 1)
42  { cerr << "Error: X509_STORE_add_crl returned " << ret << "\n" << ERR_error_string(ERR_get_error(), NULL) << "\n"; exit(1); }
43  ret = X509_STORE_set_flags(store, X509_V_FLAG_CRL_CHECK);
44  if(ret != 1)
45  { cerr << "Error: X509_STORE_set_flags returned " << ret << "\n" << ERR_error_string(ERR_get_error(), NULL) << "\n"; exit(1); }
46
```

Digital Signature

UNIVERSITÀ DI PISA

Write a "certvfy" program, which impersonates the client and:

CP! 3) Loads and verifies the server's certificate; outputs the outcome, the subject's distinguished name and the CA's distinguished name.

15 minutes

Client Checkpoint 3

UNIVERSITÀ DI PISA

```
// load the peer's certificate:
string cert_file_name;
cout << "Please, type the PEM file containing peer's certificate: ";
getline(cin, cert_file_name);
if(!cin) { cerr << "Error during input\n"; exit(1); }
FILE* cert_file = fopen(cert_file_name.c_str(), "r");
if(!cert_file){ cerr << "Error: cannot open file '" << cert_file_name << "' (missing?)\n"; exit(1); }
X509* cert = PEM_read_X509(cert_file, NULL, NULL, NULL);
fclose(cert_file);
if(!cert){ cerr << "Error: PEM_read_X509 returned NULL\n"; exit(1); }

// verify the certificate:
X509_STORE_CTX* certvfy_ctx = X509_STORE_CTX_new();
if(!certvfy_ctx)
{ cerr << "Error: X509_STORE_CTX_new returned NULL\n" << ERR_error_string(ERR_get_error(), NULL) << "\n"; exit(1); }
ret = X509_STORE_CTX_init(certvfy_ctx, store, cert, NULL);
if(ret != 1)
{ cerr << "Error: X509_STORE_CTX_init returned " << ret << "\n" << ERR_error_string(ERR_get_error(), NULL) << "\n"; exit(1); }
ret = X509_verify_cert(certvfy_ctx);
if(ret != 1)
{ cerr << "Error: X509_verify_cert returned " << ret << "\n" << ERR_error_string(ERR_get_error(), NULL) << "\n"; exit(1); }

// print the successful verification to screen:
char* tmp = X509_NAME_oneline(X509_get_subject_name(cert), NULL, 0);
char* tmp2 = X509_NAME_oneline(X509_get_issuer_name(cert), NULL, 0);
cout << "Certificate of \"" << tmp << "\" (released by \"" << tmp2 << "\") verified successfully\n";
free(tmp);
free(tmp2);
```

Digital Signature

UNIVERSITÀ DI PISA

Write a "certvfy" program, which impersonates the client and:
4) If verification went well, loads "file.txt" and "file.txt.sgt".
CP! 5) Verifies the signature over the loaded file.

10 minutes

Client Checkpoint 5

UNIVERSITÀ DI PISA

```
122    // create the signature context:
123    EVP_MD_CTX* md_ctx = EVP_MD_CTX_new();
124    if(!md_ctx){ cerr << "Error: EVP_MD_CTX_new returned NULL\n"; exit(1); }
125
126    // verify the plaintext:
127    // (perform a single update on the whole plaintext,
128    // assuming that the plaintext is not huge)
129    ret = EVP_VerifyInit(md_ctx, md);
130    if(ret == 0){ cerr << "Error: EVP_VerifyInit returned " << ret << "\n"; exit(1); }
131    ret = EVP_VerifyUpdate(md_ctx, clear_buf, clear_size);
132    if(ret == 0){ cerr << "Error: EVP_VerifyUpdate returned " << ret << "\n"; exit(1); }
133    ret = EVP_VerifyFinal(md_ctx, sgnt_buf, sgnt_size, X509_get_pubkey(cert));
134    if(ret == -1){ // it is 0 if invalid signature, -1 if some other error, 1 if success.
135       cerr << "Error: EVP_VerifyFinal returned " << ret << " (invalid signature?)\n";
136       exit(1);
137    }else if(ret == 0){
138       cerr << "Error: Invalid signature!\n";
139       exit(1);
140    }
141
142    // print the successful signature verification to screen:
143    cout << "The Signature has been correctly verified! The message is authentic!\n";
144
145    // deallocate data:
146    EVP_MD_CTX_free(md_ctx);
147    X509_free(cert);
148    X509_STORE_free(store);
149    //X509_free(cacert); // already deallocated by X509_STORE_free()
150    //X509_CRL_free(crl); // already deallocated by X509_STORE_free()
151    X509_STORE_CTX_free(certvfy_ctx);
```