Network Security: Lab#2

Objectives

- To learn to use message digests
  - MD5
- To learn to use secure hash functions
  - SHA-1, SHA-2
- To learn to use one of the public-key cryptography standards
  - RSA
- To learn to use digital signatures

Libraries Used in this Lab

- OpenSSL: an open source implementation of SSL and TLS protocols
  - Widely used on various platforms
    - UNIX-like: Linux, Solaris, Mac OS X, BSD
    - Windows
  - Cryptographic algorithms supported
    - MD5, SHA-1, SHA-2
    - RSA
  - (Installation skipped: the same as in Lab#1)

OpenSSL Command-Line Tools

- OpenSSL command-line tool
  - MD5: *openssl md5 <file>*
  - SHA: (-224, -256, -384, -512)
    - SHA: *openssl sha <file>*
    - SHA-1: *openssl dgst -sha1 <file>*

OpenSSL Command-Line Tools

- Alternative commands
  - MD5: *openssl dgst -md5 <file>*
  - SHA:
    - SHA: *openssl dgst -sha <file>*
    - SHA-1: *openssl dgst -sha1 <file>*

Experiment Scenario - Encryption

- Prepare a file to be encrypted, say “original.txt”
- First, we encrypt the file into encrypted file “enc.txt” (using receiver’s public key)
- Then, we decrypt it and get another file “dec.txt” (using receiver’s private key)
- Finally, we check if the decrypted file is the same as the original file

Another Scenario – Digital Signature

- Prepare a file to be signed, say “original.txt”
- First, we encrypt the file into encrypted file “sig.txt” (using sender’s private key)
- Then, we decrypt it and get another file “des.txt” (using sender’s public key)
- Finally, we verify if the decrypted file is the same as the original file

Yet Another Scenario – both

- Prepare a file to be signed and encrypted, say “original.txt”
- First, we encrypt the file into encrypted file “sign.txt” (using sender’s private key)
- Then, we encrypt the file into encrypted file “enc.txt” (using receiver’s public key)
- Third, we decrypt it and get another file “dec.txt” (using receiver’s private key)
- Four, we decrypt it and get another file “des.txt” (using sender’s public key)
- Finally, we verify if the decrypted file is the same as the original file

Limitations of the Tool

- File size: cannot be larger than 64 bytes
  - It depends on the key size (default: 512 bits)
  - We can set a larger key size
    - E.g. for generating private key of 1024 bits:
      - *openssl genrsa -out key.s 1024*
  - Still a serious limitation

Real Application Scenario

- For each person, a pair of keys is first generated
- To send a large file, it can be encrypted in two possible ways
  - The file must be first divided into pieces and encrypted by RSA
  - The file can also be encrypted by a session key using AES, in which the session key is encrypted by RSA
- The decryption process is also followed by a merge of decrypted pieces or by session key

OpenSSL Command-Line Tools

- OpenSSL command-line tool for RSA
  - Notation:
    - <sk>: secret key
    - <pk>: public key
  - Key management:
    - Private key generation: *openssl genrsa -out <sk>*
    - To output public key: *openssl rsa -in <sk> -pubout -out <pk>*
  - Encryption/decryption:
    - Encrypt with public key:
      openssl rsautl -encrypt -pubin -inkey <pk> -in <f1> -out <f2>
    - Decrypt with private key:
      openssl rsautl -decrypt -inkey <sk> -in <f2> -out <f3>

OpenSSL Command-Line Tools

- Alternative commands for RSA: (using *pkeyutl*)
  - Encryption/decryption:
    - Encrypt with public key:
      openssl pkeyutl -encrypt -pubin -inkey <pk> -in <f1> -out <f2>
    - Decrypt with private key:
      openssl pkeyutl -decrypt -inkey <sk> -in <f2> -out <f3>

OpenSSL command-line tool for Digital Signature

- Notation:
  - <ori>: original file
  - <sig>: signature file
- Digital signature:
  - Sign: *openssl rsautl -sign -inkey <sk> -in <ori> -out <sig>*
  - Verify: *openssl rsautl -verify -pubin -inkey <pk> -in <sig>*
- Alternative commands:
  - Sign: *openssl pkeyutl -sign -inkey <sk> -in <ori> -out <sig>*
  - Verify: *openssl pkeyutl -verify -inkey <sk> -sigfile <sig> -in <ori>*
  - Recover: *openssl pkeyutl -verifyrecover -pubin -inkey <pk> -in <sig>*

OpenSSL Libraries for Message Digests

- OpenSSL crypto library
  - MD5: 128-bit
    - #include <openssl/md5.h>
    - Initialize a MD5_CTX structure: MD5_Init()
    - MD5_Update()
    - MD5_Final()
    - Computes the message digest: MD5()

OpenSSL Libraries for Secure Hash Functions

- SHA1: 160-bit
  - #include <openssl/sha.h>
  - Initialize a SHA_CTX structure: SHA1_Init()
  - SHA1_Update()
  - SHA1_Final()
  - Computes the message digest: SHA1()
- EVP: high-level interface to cryptographic functions
  - #include <openssl/evp.h>
  - EVP_Digest…() functions: message digests

OpenSSL Libraries for Public-Key Cryptography

- RSA:
  - #include <openssl/rsa.h>
  - #include <openssl/engine.h>
  - RSA structure
  - RSA_...() functions
- EVP: high-level interface to cryptographic functions
  - #include <openssl/evp.h>
  - EVP_Seal…(), EVP_Open…(): public key encryption/decryption
  - EVP_PKEY…() functions: asymmetric algorithms

OpenSSL Libraries for Digital Signature

- EVP: high-level interface to cryptographic functions
  - #include <openssl/evp.h>
  - EVP_Sign…(), EVP_Verify…(): digital signature

Summary

- Key generation
- Encrypting a file (using public-key cryptography)
- Decrypting a file (using public-key cryptography)
- Signing a file
- Verifying a file
- Generating and verifying message digests