EHN 410

Practical 1 Lecture

Mr. Tapfuma Shang Chanaiwa

University of Pretoria

February 8, 2019

Outline

General Information

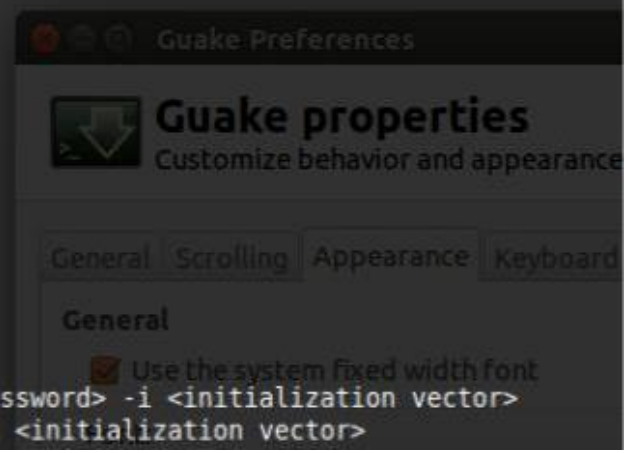
- 3 Members per group (register on ClickUp)
- All practicals need to be implemented in Linux (Ubuntu, Mint, etc.)
- Absolutely no hard-coding! (25% penalty if I have to wait for you to modify your code)
- Ensure that you are able to demonstrate different parts of your practical implementation in case your code fails (without recompiling your code)
- Verify that your encryption algorithms can encrypt/decrypt files from other implementations (will be important in practical 2 and 3)

Code execution

Command-line parameters example

```
Usage:
1.      -m <mode> <text to decrypt> <password> <initialization vector>
2.      -e <mode> <input file> <encrypted file>  <password> <initialization vector>
3.      -d <mode> <encrypted file> <descrypted file> <password><initialization vector>
4.      -brute <mode> <encrypted file> <decrypted file>  <initialization vector>
5.      -dict <mode>  <encrypted file> <decrypted file> <dictionary file> <initialization vector>


Parameters:
        -m <mode>
        -t <text to decrypt>
        -p <password>
        -i <initialization vector>
        -e (select encryption of a file)
        -fi <input file>
        -fo <output file>
        -d (select decryption of a file)
        -brute (brute force decryption
        -dict (dictionary decryption
Example usage:
1.      -d -m <mode> -fi <encrypted file> -fo <decrypted file> -p <password> -i <initialization vector>
2.      -brute -m <mode> -fi <encrypted file> -fo <decrypted file> -i <initialization vector>
```



Software Requirements

Code libraries

- Openssl (Creation of certificates)
- libssl (Binaries for linking)
- libssl-dev (Header files)
- libssl-doc (User manual)
- Doxygen (Documentation generation)
- Doxygen-doc (User manual)

X.509 Certificates

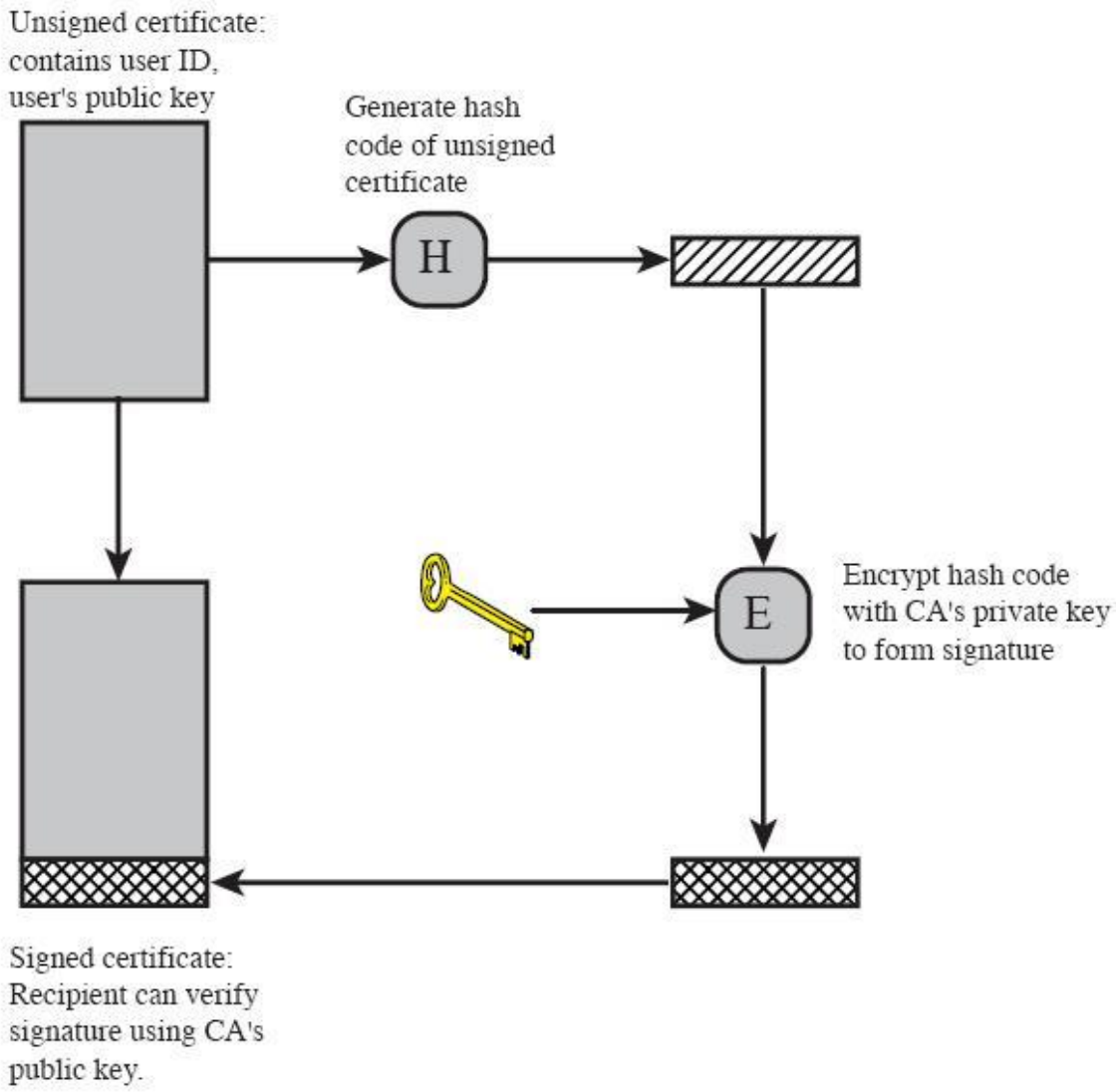
- Makes use of public-key cryptography to provide authentication/identification to users.
- Used in IP security and SSL/TLS.
- Certificates are created by some trusted Certification Authority (CA).
- Web browsers have these CA certificates "built-in".
- Web servers using HTTPS need to have their certificates signed by a CA at a certain cost.
- Large organisations may implement their own CA if they have many servers using SSL.
- Refer to Chapter 4 for more information.

Example: http://www.up.ac.za/certificates

Generating Certificates

1. CA generates public/private key pair.
2. CA generates certificate containing public key and distributes it.
3. Web server generates public/private key-pair.
4. Web server generates certificate signing request (CSR).
5. CA signs CSR.
6. Client can choose to trust the web servers certificate.

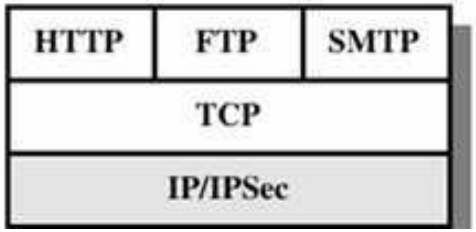
Generating Certificates

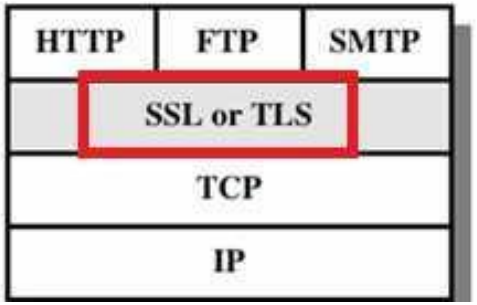


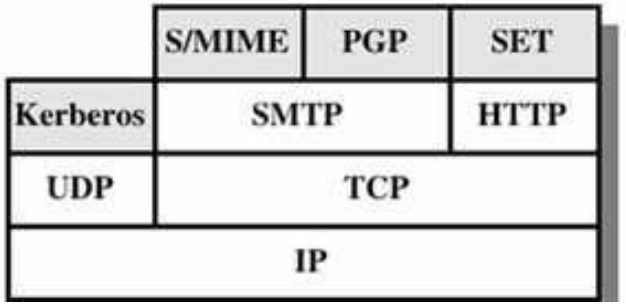
SSL/TLS

- Is a general purpose service that relies on TCP.
- Provides security services to higher-layer protocols such as HTTP.
- Requires handshaking to establish connection.
- Chapter 6 provides more background on SSL and TLS.

SSL/TLS in the TCP/IP stack







OpenSSL

- Is a popular open source SSL library.
- Supports TLS and SSL v2/v3.
- Supports various encryption/decryption cyphers.

Creating Sockets:

- Make use of OpenSSL's BIO library
- Provides an abstraction to create either normal- or secure-sockets.
- See practical guide for more detail.

Server Implementation

- Should be implemented in C (not C++)
- Should be able to handle links in a web-page (i.e. multiple pages/tabs in the web-browser)
- Must be able to handle multiple client requests (multi-threaded). (Hint: make use of pthread library) .
- Must be able to transfer large files from server to client.
- Must be able to transfer various file-types from server to client (e.g. .log , .mp3 , .aux, etc.)
- Hint: Investigate how web-browsers send requests through the GET command.

Client Implementation

- The client should be able to connect to your SSL server.
- It must be able to verify if the server's certificate is valid.
- Must be able to request web-pages and files.
- Does not need to render the page, only display the source in the terminal.

Doxygen

- Library used for software documentation.
- Easily generates HTML and or Latex output of your code.
- Can automatically generate dependency graphs of your code.
- Can incorporate marked-down (.md) file formats into documentation.
- Has a certain predefined structure for comments.
- Be sure to include adequate comments in your code to explain each code section (every 5-10 lines).
- See Doxygen man-pages for more details.

Deliverables

Demo

- Date: 28 February 2019 in the Net Labs
- HTTPS connection to SSL server using a Web browser (Firefox/Chrome/etc.).
- Multiple clients connecting to SSL server using your client program.
- Doxygen documentation.

Deliverables

Submission

- Doxygen documentation (as a PDF).
- Code and makefile
- HTML Files
- Certificates
- "ReadMe" file containing usage instructions.
- Upload as ZIP file where the file-name should be:

  EHN410-GXX -P1

  Where:
  - GXX your group number.
  - P1 refers to practical 1.