Inf-2301: Assignment 2

Submission deadline: 23:59 Friday October 9 2015

1 RTFM

For the Inf-2301 assignments, we are not going to provide you with complete information about protocols and the tools you need to use in order to answer all questions and write your code. You will be expected to nd and read the documentation that is widely available on the Internet or on the machines in the student lab (in the form of manual pages for example). Your fellow students can also be a great source of general information. When the information you have collected is not clear enough or you get stuck in your search for relevant documentation, you can always ask for assistance from the Inf-2301 staff. Until then, the following assistance to RTFM may be of help.

2 Manual pages

If you have never used the **man** command before, it is time to start now. The **man** command on Unix-like systems will give you a manual page for most commands and tools available on the system, for example: **man pwd** gives you an explanation of the **pwd** command. Another source of information (sometimes better than manual pages on GNU/Linux systems) are the so-called info pages. Try for example: **info pwd** for the info page on the **pwd** command. Reading manual pages effectively is something you have to learn, and spending some time 'getting used to them' typically pays itself back later. Reading the following manual pages are a good start: *man*, *apropos*, *whatis*, *whereis*, *locate* and the introduction pages for each manual section.

2.1 Internet Standards

Another important source of information you will need to consult regularly are the actual Internet standards produced by the IETF (The Internet Engineering Task Force) and other organizations. Each distinct version of an Internet standards-related specication is published as part of the "Request for Comments" (RFC) document series. This archival series is the offcial publication channel for Internet standards documents and other publications of the Internet Architecture Board (IAB) and the Internet Engineering Steering Group (IESG), and Internet community. RFCs can be obtained from a number of Internet hosts using anonymous FTP, the World Wide Web, and other Internet document-retrieval systems. If you are interested in the Internet standards process itself or just want a better understanding of the terminology used in them, then RFC 2026[1] may be of interest. Also, the IETF newcomer's presentation[2] or the "Tao of the IETF"[3] give a good idea of the standardization process and the organizations involved.

[1] Search for RFC at http://www.ietf.org/rfc.html (just type in the RFC number)
[2] http://www.ietf.org/proceedings/07dec/slides/newcomer-0/sld1.htm
[3] http://www.ietf.org/tao.html

2.2 Other Sources of Information

Often overlooked nowadays (because you cannot click on them?), but extremely valuable sources of information are libraries. A library is also something you need to learn how to use, so it never hurts to 'just' spend some time in one to get familiar with what treasures can be found there. If you prefer to click on things and stare at computer screens you can turn to a digital library. An excellent example is the digital library[4] of ACM (Association for Computing Machinery). At the welcome page of the ACM Digital Library you will see "University of Tromso" when accessing this resource from computers on campus. Another good source is IEEE Xplore[5]. At the welcome page of IEEE Xplore you should see the message "Access provided by: UNIVERSITY OF TROMSE" when accessing this resource from computers on campus. Be aware that both digital libraries will give you only limited access if your computer is not recognized as a campus computer. This is services that our university library pays for, and access is granted based on IP-addresses.

3 Introduction

In this assignment we will implement and analyze a file-sharing solution. In the implementation we will use both symmetric and asymmetric encryption. You should use AES for symmetric encryption and RSA for asymmetric encryption. In the solution you have to decide on key sizes, modes of operation, and type of padding. This might be influenced by what the crypto libraries or modules you are using supports.

There are no precode included for this assignment so you are required to deliver both Server and Client implementation yourself. Communication between the server and the client should be implemented using the socket interface (or similar, depending on your choice of programming language).

4 Part A: File sharing

Implement a server and a client that can be located at different computers. The server is launched (started) with at least one argument, the name of the file that it is providing to clients. When the server is started and has done some preparations it should wait for a request from a client. When a client sends a request the server should share the file with the client (transfer it to the client). The challenge is that the file has to be encrypted when transferred over the network. To summarize, the server has to read the content of the file from its local disk, encrypt it, and transfer the encrypted file (ciphertext) to the client. The client has to receive the file, decrypt it and write the decrypted (plaintext) file to its local disk.

The above description is somewhat simplified. This is the detailed steps of the server and the client. We start with *the server*:

1. The server is launched with the file name as an argument.
2. The server generates a fresh symmetric AES key and creates a cipher (that can performs encryption using the new AES key).
3. The server waits for requests from a client.

8. When a request arrives, the server encrypts a copy of the fresh AES key with the public key of the client.
9. The server then encrypts the file content with the AES key.

[4] http://portal.acm.org/dl.cfm
[5] http://ieeexplore.ieee.org/

10. Finally, the server transfers the encrypted file *and* the encrypted AES key to the client.

And this is *the client*:

4. The client is launched with the IP-address of the server as an argument.
5. The client generates an asymmetric RSA key-pair (or it uses a previously generated key-pair).
6. The client sends a request to the server. The request includes its public key (the public part of the RSA key-pair).
7. The client waits for the response from the server.
11. The client receives the encrypted file *and* the encrypted AES key.
12. The client decrypts the AES key with its private key.
13. The client decrypts the file content with the decrypted AES key.
14. Finally, the client writes the decrypted file to the local disk.

Some details are not described above and you should make reasonable choices when implementing the server and the client. Examples include (but are not limited to):

- How does the server (and the client) choose port number to establish the connection?
- How does the client know the name of the file (and is it necessary for it to know it)?
- How is error handling implemented (and what kind of errors should you be aware off)?

It is smart to implement this step-by-step. Start with file-transfer without encryption. Then encrypt the file (and not the keys). Finally you introduce the encryption of the AES key with RSA encryption.

5 Part B: Analyze and redesign

In part B you are not going to implement anything (no code to write). We should however analyze the implementation from part A and suggest improvements (and redesign) to this solution. Below a few questions are listed, but you do not have to answer them sequentially. It is better if you manage to write a general text that answers all the questions. Your answers should be discussed and argued for.

1. Discuss your solution in part A based on C. A. I. (Confidentiality, Integrity, Availability) and A. A. A. (Assurance, Authenticity, Anonymity). Give examples of possible threats your solution can be exposed to. What are these threats attacking (related to C. A. I. and A. A. A.)? How thus these threats influence the application for your solution (in what scenario can it be used)?

We will now focus on three of the terms from C. A. I. and A. A. A.

2. Discuss how we can achieve confidentiality, integrity and authenticity in the solution implemented in part A. This discussion should include a description of a possible redesign (extension) of the solution and assumptions that must hold for the redesigned solution to work. Hint: signatures, certificates and certificate authority (CA).

3. Certificates are bound to trust and a third part. Explain how all this is connected and how we can see it in your redesign. Who trusts who, and how is this trust used?

This discussion and analyze should be both practical (related to part A) and extensive (show that you understand and can analyze challenges and possible solutions related to security). Meaning, part B of this assignment will be at least several pages long.

6 Report

For part A describe the design and implementation of the programs you made. Do not simply copy a lot of content from external sources like RFCs, refer to them instead. Do not include the source code in the text (it should be submitted as separate files). However, the source code should be readable enough, both in structure and the way it is commented, for another programmer to verify its correctness. In other words, your report with source code needs to convince us that your programs work, not execution of the actual programs! Include also a description on how you tested your programs.

For part B you should produce an analytic text tied to your description of part A. The text should not assume that the reader know the detailed content of your text books, and it should refer to external sources used.

Include a 'discussion section' in which you note your observations on this assignment, the tools, protocols and libraries you studied, and the documentation you consulted. Maybe you observed something interesting (or just surprising to you). If so, write about it in your report.

Describe what aspects of the solution that caused you implementation headaches, if any. Also report any particular assistance you got from course staff or your fellow students that has been crucial to the fulfillment of the assignment. You are strongly advised to write documentation/notes along the way.

7 Deliverables

This assignment must be fulfilled by every individual student (no group delivery). You may use your favorite programming language, whatever that may be. Reasonable choices include C, C#, Python, Ruby and Go.

Please follow these guidelines for the delivery of your result. Create a directory named after your UiT user name (e.g. abc012). This directory should contain all your deliverables. Assignment documentation and report can be composed in any program, but must be delivered either in adobe acrobat format (pdf) or postscript (ps).

For source code, create a sub directory called `src`. This directory should contain a README file in plain ASCII text, describing how to build (compile) your project, install it (if needed) and how to run it (arguments, etc).

The entire directory tree should be archived in a tar/gzip or zip file. This file should be named as the assignment directory (with the appropriate `.tar.gz`, `tgz` or `.zip` suffix). Finally use Fronter to upload this file to the "Inf-2301 assignment 2" hand-in folder in the Inf-2301 room.