CS 4389 Introduction to Data Security

Programming Project

Due on November 30th 2012

Security guarantees of most services on the web rely on passwords set by the users. In an effort to encourage strong passwords, most systems disallow short passwords; require that passwords contain letters, numbers and punctuation characters; discard old passwords periodically and etc. However, sooner or later users –unable to cope with so many supposedly random sequences– get frustrated with such strategies and write down the passwords for easy lookup. The end result is increased risk of security breaches.

In this project, we will remedy the problem using the security constructs we have learned throughout this class. Specifically, you will write a program that provides confidentiality and authenticity of the password file.

Suppose all your passwords are stored in a textual file. Confidentiality ensures that no one reads the file. This requires that you encrypt your file with some encryption function. The purpose of authenticity is for you to find out if your encrypted data has been altered with. This requires that you generate message authentication codes (MACs) from the encrypted file. Since confidentiality and authenticity are guaranteed, your passwords will be safe.

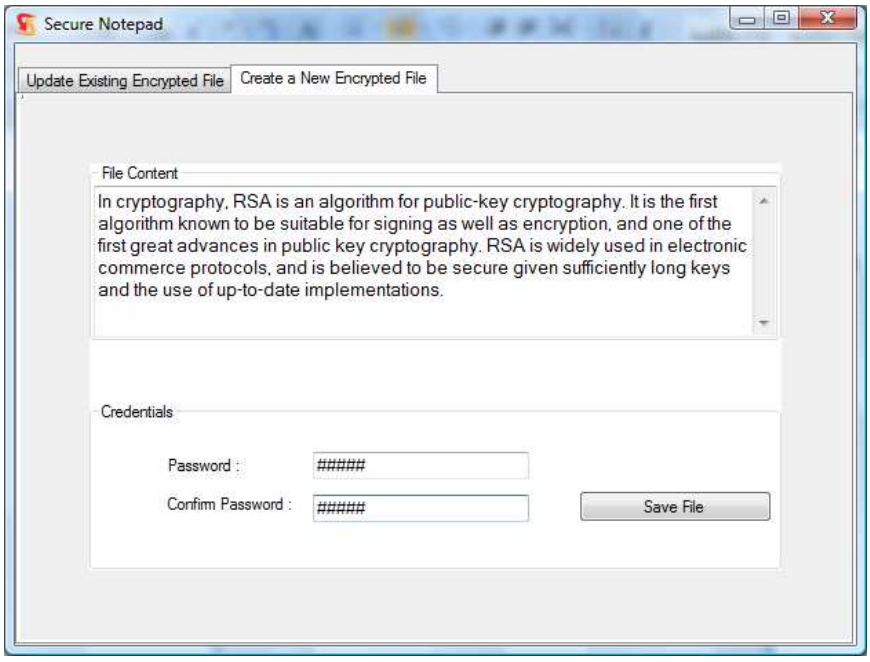

Figure 1. Sample GUI for the program

We describe the details on our implementation. The interface for creating new encrypted files will be similar to Figure 1. The user will fill in a text box, and then provide credentials for the encrypted file. Finally, when the “Save File” button is hit, a pop-up window will ask for a filename and a path and create the file accordingly. First few bytes of this file will be the MAC and the rest will be the encrypted file. You may use a separator to separate the two.

Anybody that opens the encrypted file (named “test.spad” in Figure 2) will read a bunch of meaningless characters.

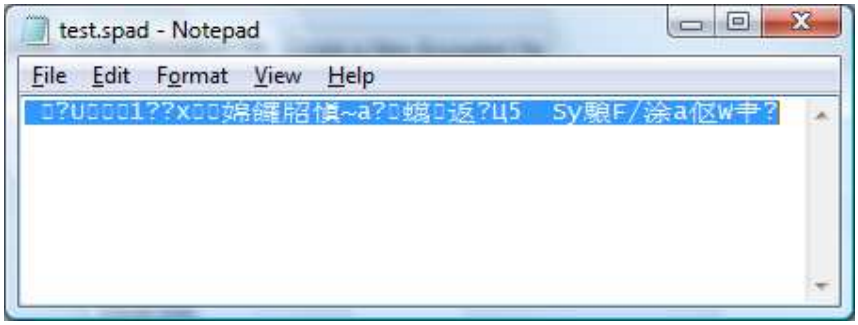

Figure 2. Contents of the encrypted file

Later, when the user wants to access or modify an encrypted file, the program is run again. First, the user selects an encrypted file by providing the path. Once the file is selected, MAC needs to be validated. If the file has been altered with, the user should be informed with a proper error message. Otherwise, the encrypted file will be decrypted and displayed in an editable textbox. After finished reading or writing the file, the user can save or discard the changes. Notice that the changes might include updating the credentials as well. See Figure 3 for details.

1. Project Requirements

- Use AES as the encryption function and HMAC for generating the MAC.
  - See below for generating the keys from credentials.
- Use any programming language you want.
- Always keep decrypted files in memory rather than a temporary file.
- Demonstrate your work.
  - Demo times will be announced later.

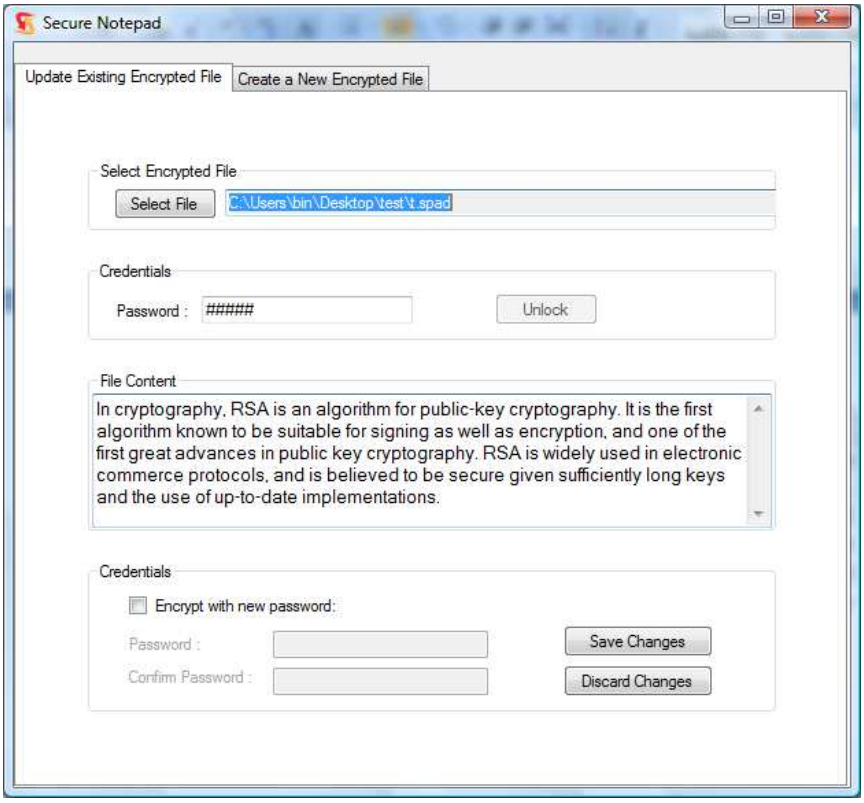

Figure 3. Reading, editing the encrypted file

2. Submission Guidelines

You may work in groups of size up to 3. Please submit your code and a report that is maximum four pages long, single spaced, 11 font size with at least 1 inch margins on all sides. All submissions should be through Elearning.

In the project report discuss the following:

- Major steps of the implementation,
- The difficulties you faced during implementation,
- The division of labor among group members,
- Usage of the software with screenshots.

3. Generating AES and HMAC passwords from credentials

The following mechanism called *salting* can be used to generate two passwords from the credentials. Let c denote the credentials and h denote a cryptographic hash function. Given a bit string (i.e., salt) s_1 and a non-zero integer value r, the AES key K_1 can be computed as follows:

$x_0 = 0$ and
$x_i = h(x_{i-1} \| c \| s_1)$ for $i = 1, ..., r$
$K_1 = x_r$

To compute the HMAC key K_2, simply repeat the same algorithm with another salt s_2. Here r, s_1 and s_2 should be hardcoded in your program and never be changed.

Notice that the method described above requires that $|K_1| = |K_2|$ = block size of h. So choose the hash function h based on the key lengths you want for AES and HMAC.

4. Java Example

You can find more info on Java cryptography extension by using the following link.

http://java.sun.com/j2se/1.4.2/docs/guide/security/CryptoSpec.html