Project: Dropbox - Part 4

CS131: Fundamentals of Computer Systems

Tue., May 7, 2013

1 Introduction

In the fourth part of this project, you will modify your existing Dropbox program to have some additional security measures. You will be adding 4 measures: message authentication, message integrity, message confidentiality, and server-side encryption of data. All of these will involve using cryptographic primitives discussed in class.

2 Carrying on from Part 3

We will not be grading your functionality from previous parts of the project, although you should make sure that your implementation from previous parts has no fatal bugs, as that will make grading your part 4 harder.

3 Specification

3.1 Message Security: Definitions

Message authentication means that the receiver of the message can validate the identity of the sender of the message. Message integrity means that the receiver of the message validate that the contents of the message have not been modified by anyone except the sender of the message. Message confidentiality means that no one other than the sender and receiver of the message can read the contents of the message. To actually provide these security guarantees in the real world would require many extra steps, but the measures you implement here should provide an idea of how a real world system (such as Dropbox) would work.

3.2 Establishing a Connection

Instead of designing or implementing a handshake protocol (which needs to be done very precisely to actually be secure), we will be manually transferring the shared keys between the clients and server. Essentially, we are depending on a human messenger to transmit this key rather than a handshake protocol (although if you are interested in what one would look like, read the section in the book that describes Diffie-Helman key exchanges). Your server should create unique keys for encryption, authentication, and for server-side file encryption, for a total of 3 types keys. The client should only need two keys, the authentication key and the encryption key, in order to start a connection with the server. You have two options to ensure unique client connections: create unique encryption keys for each client (in which case you do not need separate authentication keys, although you will need to use checksums to verify message integrity), or you can have a common encryption key and use unique authentication keys for each client. Your server should print out some string representation of these keys, so that these can be passed into the clients as a command line argument. The command line argument can be a random seed of some kind that is used to initialize each key, or it could be some serialized information that can be used to create the key. You may use either symmetric keys or asymmetric keys.

3.3 1, 2: Message Authentication and Integrity

Message integrity and message authentication go hand-in-hand, as this will be accomplished by creating a message authentication code or MAC (see the textbook for more details) that the receiver can use to verify that the message was signed by someone using the same key, and that the same message was signed by that person as the message that was received.

3.4 3: Message Encryption

Message encryption will be accomplished much as might be expected. Every message transferred between the client and server must be encrypted before being transmitted over the network, and decrypted before it can be read by the recipient of the message.

3.5 4: Server-side Encryption of Data

Your server should encrypt all of the data it holds before writing it to the file system, this way all files it stores are not stored raw. Your server should use a key that is not used anywhere else to accomplish this, so that the data can only be accessed by your server program, and not by anyone else (including the clients) directly.

4 Java Cryptography

More information on the Java cryptography classes can be found here `http://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html`. The `KeyGenerator`, `Cipher`, and `Signature` classes in particularly could be useful.

5 README and Commenting

You will have to update your README for this part of the project, again documenting any high-level design decisions made and any changes made to your program's structure. Please also document how your program may differ because of any bugs in your part3 implementation, if it does at all.

6 Handin

To hand in your database implementation, run

```
cs131_handin dropbox-part4
```

from your project working directory. Make sure you include all .java files and your README.

If you wish to change your handin, you can do so by re-running the handin script. Only your most recent handin will be graded.