CS105 – Computer Systems Spring 2024

Assignment 4: Cryptography

Due: Tuesday, February 27, 2024 at 11:59pm

This assignment should be completed with a partner. You should submit one solution to Gradescope.

For this assignment, you will implement some core pieces of a secure instant-messaging application, as well as simulate a Dolev-Yao attack against that application.

1 Implementation

Your task is to build a collection of four programs named Alice, Bob, Mallory, and Gen:

- **Alice:** repeatedly prompts the user for a string then sends that string over the network to the recipient.
- **Bob:** displays strings that are received from the sender.
- **Mallory:** the Dolev-Yao attacker. Mallory receives a message as input from the network, displays the message, and prompts the user whether to forward the message as is, to modify the message before forwarding it, or to drop the message and not forward it. Mallory can also store and replay old messages. The sophistication of modification and replay that you implement is up to you, as long as you are able to carry out the demo described below.
- **Gen:** generates public–private key pairs and stores them in files.

To help you get started, I've provided simple starter code that sends messages over a TCP connection from Alice to Bob in both Python and Java.

Requirement 1: Network communication. Alice, Bob, and Mallory communicate with one another over TCP. The communication architecture is that Alice sends messages to Mallory, who sends messages to Bob:

```
Alice --> Mallory --> Bob
```

Hostnames (or network addresses) and ports must not be hardcoded. They could be specified as command-line arguments, or accepted as program inputs.

The expected workflow is as follows:

1. Run Gen to generate key files, which are then manually distributed to the filesystems from which Alice, Bob, and Mallory will run. Alice receives her public and private keys as well as Bob's public keys. Bob receives his public and private keys as well as Alice's public keys. Mallory receives both Alice and Bob's public keys, but not their private keys.
2. Start Bob.
3. Start Mallory, specifying Bob's address as the recipient.
4. Start Alice, specifying Mallory's address as the recipient.

5. Type strings into Alice, which are sent over the network to Mallory. Use Mallory to read, modify, and/or delete messages. Messages sent by Mallory to Bob should be displayed by Bob. If Bob is able to detect Mallory's actions on messages, then a notification about this action should be displayed.

Requirement 2: Cryptography. At startup, the system should provide the ability to operate in each of four configurations:

1. No cryptography: messages are not protected.
2. Symmetric encryption only: the confidentiality of messages is protected.
3. MACs only: the integrity of messages is protected.
4. Symmetric encryption then MAC: both the confidentiality and integrity of messages is protected with Enc-then-MAC.

Once an instance of the system is started, the configuration need not be changeable. The configuration must not be hardcoded. It could be specified as a command-line argument, or accepted as a program input.

Since Alice and Bob do not initially share any symmetric keys, you need a key establishment protocol; use the protocol provided in the appendix of this writeup.

The system configuration should determine how Mallory interprets and displays messages. We ask that you make Mallory's display easy for a human to interpret. More specifically, for each cryptography mode, Mallory should display the following:

1. No cryptography: Mallory displays the same plaintext message that Alice sent. This should be the same as the string entered at Alice—not (e.g.) some human-unreadable byte array.
2. Symmetric encryption only: Mallory displays the ciphertext.
3. MACs only: Mallory displays both the plaintext message and the tag.
4. Symmetric encryption then MAC: Mallory displays both the ciphertext and the tag.

Requirement 3: Interface. Your program should run on the command line. It may take additional configuration arguments as program input or command line arguments. The messages should be accepted as program input.

Implementation: Choose a programming language with which you are comfortable and that has library support for the cryptographic and networking functionality you will need. Java and Python are both good choices. If you are intend to use a language other than Java or Python, please discuss this with me before you begin so that we can ensure that there are appropriate crytography libraries and that it can be run on one of the approved systems. You may find it saves you time later if you use the same language a you plan to do your project it. Use library implementations of encryption schemes, block cipher modes, MACs, digital signatures, and hash functions. Do not use library implementations of higher-level protocols, such as SSL/TLS.

Underspecification: All further details are up to you. Make reasonable choices and be prepared to defend them.

2 Attacks

Once you have finished the implementation, add functionality to Mallory that enables Mallory to (attempt to) execute each of the following attacks:

1. Read the contents of a message
2. Undetectedly modify a message
3. Undetectedly replay an old message

Whether these attacks will succeed will depend on the current configuration (and potentially on your implementation choices).

3 README

Include a README file that contains:

1. Detailed setup. This should include a list of line-by line instructions for installing any required dependencies needed to run your system on the course VM or my Mac. (It should also be clear which system your solution will run on!)

2. Detailed Usage. This should include a list of line-by line instructions for compiling (if necessary) and running the executables to perform the experiments listed above

3. Rationale: Explain the use of cryptography in your system. The document should include protocol narrations to document the cryptographic protocol used in each of your four configurations. The document should also provide a justification for your choices of key lengths and cryptographic algorithms.

Specification. This part of the assignment is deliberately underspecified. Detail and justify your choices here.

External libraries. If you use any external libraries, list them here along with what your code uses them for.

Known problems. Detail any known problems with your rationale, specification,or implementation. You may also include anything else that you see fit.

4 Feedback

In the interest of improving future iterations of this course, please answer the following questions and upload them in a file called feedback.txt:

1. How long did you spend on this assignment? (Including time spent in class.)
2. Any other comments or feedback?

5 Demo

Schedule a time to demo your code for me. This should happy sometime in the week following the assignment deadline. Be prepared to demonstrate the following experiments:

1. Part 1
   (a) Run Gen to generate new key files.
   (b) Start Bob, Mallory, and Alice in ”no-cryptography” configuration.
   (c) Send messages from Alice to Mallory to Bob.
   (d) Use Mallory to delete a message.
   (e) Use Mallory to modify a message.
2. Part 2
   (a) Start Bob, Mallory, and Alice in ”Enc-only” configuration.
   (b) Send messages from Alice to Mallory to Bob.
   (c) Use Mallory to modify a message without Bob detecting the modification
3. Part 3
   (a) Start Bob, Mallory, and Alice in ”Mac-only” configuration.
   (b) Send messages from Alice to Mallory to Bob.
   (c) Use Mallory to replay an old message.
   (d) Use Mallory to delete a message and pass the next message through.
   (e) Use Mallory to modify a message.
4. Part 4
   (a) Start Bob, Mallory, and Alice in ”Enc-then-Mac” configuration.
   (b) Send messages from Alice to Mallory to Bob.
   (c) Use Mallory to modify a message.
   (d) Use Mallory to replay a message as described in your attacks.txt file

What to Submit

If you work with a partner, submit only one copy rather than submitting the same solution independently. Be sure that your README and overview document both clearly identify both partners, and be sure to tag your partner on Gradescope when you submit.

Submit the following files: (1) Your README file, (2) Your source files for Gen, Alice, Bob, and Mallory, (3) Your attacks.txt file, and (4) Your feedback.txt file.