

ISA 656, Assignment 2 *due October 9th, 11:59pm*

GENERAL INSTRUCTIONS

This homework has two parts: one part with a set of theory questions and a programming part. Programming can be done in either C or Java. Submissions should include the **annotated** source code. Programs that do not compile will get a low grade. Make sure your programs do not crash when given bad input, but rather produce a useful warning or error message and take the appropriate action (recover or quit).

SUBMISSION INSTRUCTIONS

The compressed files (either tar or zip) will be submitted using email to astavrou@gmu.edu. Please make sure that you send me a **single compressed file with all the documents and code. The compressed file should not exceed 1MB.**

THEORY/WRITTEN QUESTIONS (50 points)

1)[15 points] Firewalls:

- a) What is a proxy firewall and how is it different from a network (or transparent) firewall?
- b) What does NAT stand for, and how does the mechanism work? Describe what, if any, security NAT provides (or fails to provide).
- c) Where would you place a web server in an organization assuming that you can use a network firewall and why?

2)[20points] Assume you have the following firewall policy:

Action	prot	Source	Destination	Source port	Destination port
ACCEPT	udp	0.0.0.0/0	129.174.17.180	*	53
ACCEPT	Tcp	55.66.77.0/24	129.174.17/180	*	22
REJECT	Tcp	55.66.77.12	129.174.17.180	4500	22

ACCEPT	Tcp	127.0.0.1	129.174.17.180	*	6000
DENY	Tcp	0.0.0.0/0	129.174.17.180		6000
REJECT	udp	0.0.0.0/0	129.174.17.180	*	32768
REJECT	tcp	0.0.0.0/0	129.174.17.180	*	32769
REJECT	tcp	0.0.0.0/0	129.174.17.180	*	32768
ACCEPT	tcp	0.0.0.0/0	129.174.17.180	*	80
ACCEPT	udp	129.174.16.20	0.0.0.0/0	53	1025:65535
ACCEPT	udp	129.174.20.100	0.0.0.0/0	53	1025:65535
ACCEPT	udp	129.174.18.100	0.0.0.0/0	53	1025:65535
ACCEPT	All	0.0.0.0/0	0.0.0.0/0	*	*
REJECT	tcp	0.0.0.0/0	0.0.0.0/0	*	*
REJECT	udp	0.0.0.0/0	0.0.0.0/0	*	*
DENY	tcp	0.0.0.0/0	129.57.17.180	*	6000:6010
DENY	tcp	0.0.0.0/0	129.174.17.180	*	0:1024
DENY	All	0.0.0.0/0	129.174.17.180	*	*

a) Identify any policy conflicts or redundancies.

b) Show what would happen to the following packet under the three tie-breaking strategies (first match, best match, last match).

```
+-----+-----+-----+-----+
| TCP | s:55.66.77.12:4500 | d:129.174.17.180:22 | data... |
+-----+-----+-----+-----+
```

c) Show what would happen to the following packet under the three tie-breaking strategies (first match, best match, last match).

```
+-----+-----+-----+-----+
| UDP | s:55.66.77.12:4500 | d:129.174.17.180:22 | data... |
+-----+-----+-----+-----+
```

3) [15 Points] Assume a cryptographic algorithm in which the performance for the good guys (the ones that know the key) grows linearly with the length of the key, and for which the only way to break it is a brute-force attack of trying all possible keys. Suppose the performance for the good guys is adequate (e.g., it can encrypt and decrypt as fast as the bits can be transmitted over the wire) at a certain size key. Then suppose advances in computer technology make computers twice as fast. Given that both the good guys and the bad guys get faster computers, does this advance in computer speed work to the

advantage of the good guys, the bad guys, or does it not make any difference?

PROGRAMMING (50 points)

PART A:

Create a utility that produces an AES key from a pass-phrase provided on the command line and stores the key in a file named “.myaeskey.” If no pass-phrase is provided, a random key is generated and stored in “.myaeskey.” If the file “.myaeskey” exists, it is saved to “.myaeskey.bak.” If “.myaeskey.bak” exists, it is overwritten.

PART B:

Create a file encryption utility that uses AES in any of the four main modes to encrypt a source file. The secret key should be created using the utility from PART A. The encryption utility should both encrypt and decrypt the file. Make sure that you account for the use of IVs (for the modes that require them) and that a 'diff' of the original source and the decrypted source file succeeds.

NOTE: You do not have to implement AES yourself; you can use the libraries in either C or Java.

Make sure you document your syntax, plus any additional parameters you needed to solve the problem. Compile all your programs prior to execution.

A sample encryption/decryption session using java *might* proceed like this (the commands displayed are for Linux or Windows command prompt; in parenthesis are for windows):

```
> java createkey this_is_my_passphrase
> ls -a (or dir for windows)
    .myaeskey createkey data.txt filecrypt

> java createkey this_is_a_new_passphrase
> ls -a (or dir for windows)
    .myaeskey .myaeskey.bak createkey data.txt filecrypt

> java filecrypt -e -mode ECB data.txt
```

```
> ls -a (or dir for windows)
    .myaeskey createkey data.txt filecrypt data.txt~encrypted

> java filecrypt -d -mode ECB data.txt~encrypted
> ls -a (or dir for windows)
    .myaeskey createkey data.txt filecrypt data.txt~decrypted
> diff (or comp for windows) data.txt data.txt~decrypted

> java filecrypt -e mode CBC data.txt
> ls -a (or dir for windows)
    .myaeskey createkey filecrypt data.txt data.txt~encrypted
    data.txt~iv

> java filecrypt -d mode CBC data.txt
> ls -a (or dir for windows)
    .myaeskey createkey filecrypt data.txt data.txt~decrypted
> diff (or comp for windows) data.txt data.txt~decrypted
```

EXTRA CREDIT [15points]

Can you create a multi-threaded version of the previous program that can encrypt and decrypt multiple files at the same time? When can we benefit from using a multi-threaded approach for encryption and decryption? Are we going to have a performance gain for all the modes of AES encryption?

Resources for Java:

<http://java.sun.com/javase/6/docs/api/javax/crypto/package-summary.html>

<http://java.sun.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html>

Look for AES

Resources for C:

openssl library:

use "man openssl"

<http://www.openssl.org/>