CPSC 526 Assignment 4

Encrypted file transfer server

Geordie Tait
10013837
T02

How to compile

Client:

```
gcc -o client a4client.c -lssl -lcrypto
```

Server:

```
gcc -o server a4server.c -lssl -lcrypto
```

How to run

Client:

```
./client command file hostname:port cipher key
```

where `command` is one of `read` or `write`
`file` is the filename for the server to operate on
`hostname` is the server name or address
`port` is the server port number
`cipher` is one of `null`, `aes128` or `aes256`
`key` is the secret key

Server:

```
./server port key
```

where `port` is the port to listen on
`key` is the secret key

Correctness test

```
[taitg@csx3 a4]$ sha256sum 1mb.bin
568e7c20844b170392a0ec91b58c76e5b2b24af365e9cc4e73fecce837f3aedf  1mb.bin
[taitg@csx3 a4]$ ./client write testfile localhost:22222 aes256 key < 1mb.bin
OK
[taitg@csx3 a4]$ ./client read testfile localhost:22222 aes256 key | sha256sum
OK
568e7c20844b170392a0ec91b58c76e5b2b24af365e9cc4e73fecce837f3aedf  -
```

Communication protocol

client→server cipher,nonce (unencrypted)
(all traffic now encrypted)
server→client random challenge
client→server response computed using secret key
server→client result of authorization attempt
client→server operation,filename
server→client whether to proceed
client→server chunks of data (if writing)
server→client chunks of data (if reading)
server→client status

The **cipher** is either null, AES128 or AES256 and is specified by the client.
The **nonce** is a 16-character alphanumeric string generated randomly by the client.

These are used along with the secret key to generate **initialization vectors** and **session-keys** which are used for the remainder of the communication.

For the **challenge**, the server generates a random alphanumeric string, encrypts it with the secret key, and then sends it to the client. The client decrypts it and sends it back (the **response**), and if it does not match then the authorization fails and communication is ended. All of these messages are also encrypted by the specified cipher.

Timing tests

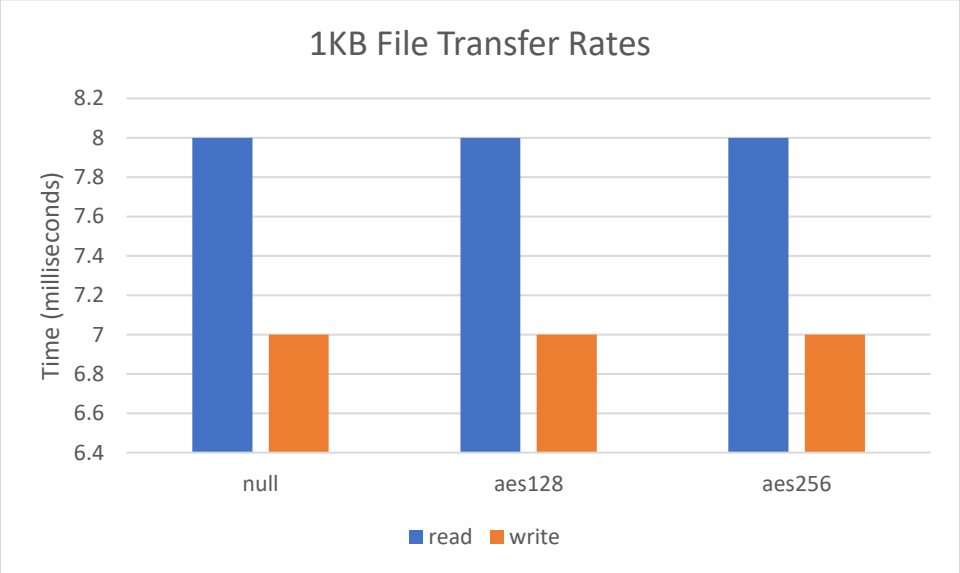

Fig 1 Time required to read or write a 1KB file with each cipher option

Fig 2,3 Time required to read or write a 1 MB or 1GB file with each cipher option

	1kb file						1mb file						1gb file					
	null		aes1		aes2		null		aes1		aes2		null		aes1		aes2	
RUN	r	w	r	w	r	w	r	w	r	w	r	w	r	w	r	w	r	w
1	25	7	8	7	9	7	26	21	33	29	37	33	13801	14356	25853	13715	30645	16965
2	8	7	8	7	8	7	22	19	32	26	36	24	15332	15465	20014	25281	27872	23211
3	8	6	7	7	7	7	22	11	33	21	35	24	15357	15175	24538	17109	27294	18871
4	8	7	9	7	8	7	22	20	33	25	37	24	15235	14886	24133	24924	24715	27029
5	8	7	8	7	8	7	22	11	32	20	36	24	13430	10900	24896	17840	22066	19479
6	21	6	8	7	8	7	22	20	33	20	35	23	13999	14021	19775	24346	28629	28039
7	8	7	8	7	8	7	23	19	32	25	35	24	12171	13390	24723	16634	28196	27593
8	8	7	8	7	8	7	22	19	33	21	36	24	13635	13456	24440	24558	28127	25274
9	8	7	8	7	8	7	23	20	34	21	35	24	13895	13948	24897	17067	23087	19702
10	8	6	8	7	8	7	22	20	33	21	36	24	14177	14014	17922	24215	24836	28185
median	8	7	8	7	8	7	22	20	33	21	36	24	13947	14018	24489	21028	27583	24243

Table 1 Raw times and medians for each test in milliseconds

Discussion

Across nearly all file sizes and modes, the read functionality takes longer than the write functionality. This is more pronounced, however, when using a non-null cipher. For a 1GB file using the null cipher, a read and a write take almost the same time.

For transferring files of size 1KB, the results show identical times across all three cipher modes, for both reading and writing (Fig 1). There is not a measurable amount of time lost encrypting the communication.

Files that are 1MB show a much greater difference in transfer speed when using non-null ciphers (Fig 2). A read using AES128 took 50% longer than one using null, and a read using AES256 took 64% longer. The difference in write times is much less drastic but it is still present.

Using non-null ciphers for 1GB files has an even greater impact on transfer speeds (Fig 3). AES128 reads took 76% longer than null ones, and AES256 reads were nearly twice as long. Null writes took just as long as reads, and for all other cipher modes they tracked closer to read times than when testing 1MB files.

These results indicate that the cipher does have a significant impact on the time it takes to transfer files. This effect is, however, dependent on the size of the file. At around 1KB, encrypting the transfer makes little difference, but around 1GB using a cipher might double the time.