

CryptServ

Deadline: October 24th, 2014.

1 Instructions

You must implement a program `cryptserv` working as follows.

`cryptserv` must take two command-line arguments; a numeric encryption/decryption key, and a filesystem path to use as Unix domain socket (`PF_UNIX/AF_UNIX`). It must then listen for connections on that socket. When a client connects, `cryptserv` must encrypt/decrypt the received data stream from that client (cf [Cryptography](#) below) and send the results back to the client. The program must support exchanging data with multiple clients simultaneously.

For extra credits you may implement the following:

- Detaching as a daemon when the command-line option `-d` is specified.
- Non-blocking I/O: communication with other clients can go on even if a client is blocked.
- Support for listening to multiple sockets/addresses.

Constraints:

- You must only depend on standard C functions (either from ISO C 1999/2011 or POSIX).
- You must not use `system` or any other mechanism that invokes an external program.

2 Cryptography

The encryption (or decryption) of the data stream from one client must use the following algorithm:

- first the C library's random generator state must be initialized (`initstate`) using the key as initial seed and a seed array of 256 bytes.
- each successive byte in the input stream must be XORed with the lowest 8 bits of successive calls to `random`:

```
char buf[256];
initstate(key, buf, 256);
setstate(buf);
out[0] = inp[0] ^ (random() & 0xff);
out[1] = inp[1] ^ (random() & 0xff);
// ...
```

Be sure that each client connection uses its own random state!

3 Example use

The following examples uses Netcat (`nc`), a quasi-standard “swiss army” network tool:

```
# in one session
$ ./cryptserv 0xdeadbeef /tmp/mysock

# in another session
$ echo hello >msg.txt
$ nc -U /tmp/mysock <msg.txt >msg.enc
$ nc -U /tmp/mysock <msg.enc >msg2.txt
$ cmp msg.txt msg.enc || echo OK
$ cmp msg.txt msg2.txt && echo OK

# in yet another session
$ nc -U /tmp/mysock </dev/urandom >/dev/null &
$ echo hello | nc -U /tmp/mysock && echo OK
```

4 Grading

- 4 points if `cryptserv` works with 1 client at a time, in constant memory space.
- +2 points if `cryptserv` works with any number of clients simultaneously, in memory space linear with the number of clients connected.
- +1 point if all of the above, and the program detaches as a daemon properly if `-d` is given.
- +1 points if non-blocking I/O is properly implemented.
- +1 points if the program can listen to multiple sockets/addresses (explain your command-line syntax in an accompanying `README` file.)
- +1 point if an enclosed `README` file contains an explanation that properly identifies the encryption algorithm used in this assignment and suggests simple improvements to make it more secure.