Gabriel Veras
CS351
12/13/2020

https://github.com/GVeras/EndToEnd-Encryption

Project Final Report

User registration

When registering the user will use the optional flag "-r" when executing the file, letting the client.py know that the user is registering instead of just logging in. The user will then type in their username, password, and role that they would like.

Under the hood, what happens is that the user will send this information to the server, and the server will then create a new text file. This text file will be storing a hash digest of the concatenated string + salt (using uuid's random id gen) using the hashlib module. The hash will then store the username, hashed password, salt and role of the user that just logged in remotely.

Users logging in (account authentication . management)

When logging in, users will send the username and password to the server. The server will then prompt them to log in again if they fail.

Under the hood, the server will be sent the username and password. The server will scan the aforementioned text file to search for the username. Once the user name is found, they will obtain the hashed password, salt, and role. The password that the user sent is then hashed with the salt in the text file, that hash digest is compared to the stored hashed password. If they are the same, then the user provided the right password and can log in remotely.

It is worth noting that the salt is not only stored with the hashed password as well, separated by "---", but also calculated in the hash function. It could have been comma separated as well.

Users connection (connecting to the right server)

Under the assumption that the server has a public key, the server could post that public key anywhere, or they could simply send it to whomever they want. For this program, I assumed that the server and client already had the matching symmetric keys (hard coded in this case). If the symmetric key was changed by even a single digit then they would not be able to connect to the server. This way, they know that they are connecting to the right server.

Secure communication

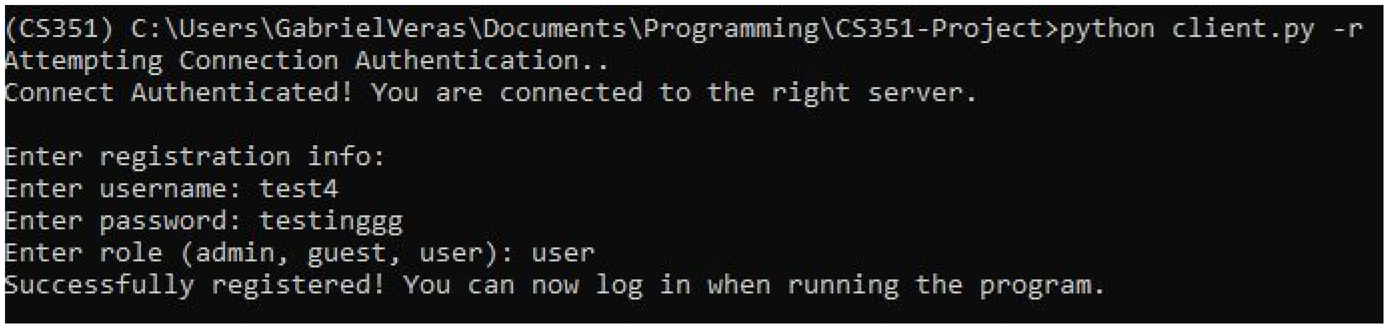
After authenticating and logging in the user will be prompted to decide if they want to be a receiver or a caller. This distinction decides on who is going to be the initiator in the communication, like who is the first to ring the other person.

The receiver will wait for a call, while the caller will be able to choose who to call based on a list of who is logged into the server at that moment. If nobody is logged on, the caller can choose to wait. Once the caller types in the name of who they are communicating with, they will then start the end to end communicating between two clients.

By this stage, the clients have already logged in and generated public / private keys for themselves. Before typing anything, the clients will have to send each other their public keys. Once that is successfully done, a status message appears showing that the communication may start and who will be the one typing first (there is turns to typing, i.e you type a message then they type a message).

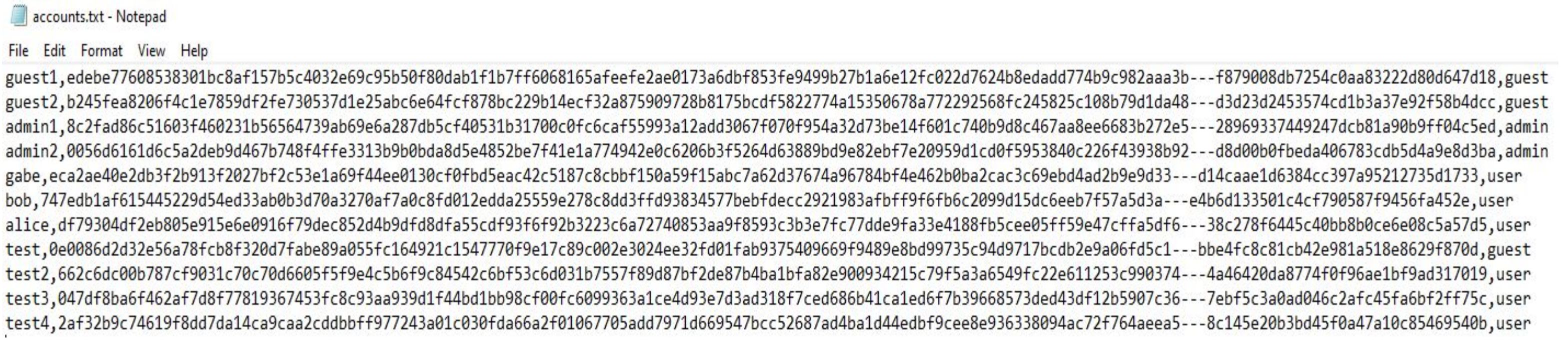
After that, the two clients will be able to send messages to each other. Assume client A sends a message, it will be encrypted using client B's public key that was just sent, and will be decrypted by client B using client B's private key. This encryption / decryption does not involve the server, as shown below. The server will only see the encrypted bytes, and will not be able to tell what the message is saying at all.

1A. Registration



```
(CS351) C:\Users\GabrielVeras\Documents\Programming\CS351-Project>python client.py -r
Attempting Connection Authentication..
Connect Authenticated! You are connected to the right server.

Enter registration info:
Enter username: test4
Enter password: testinggg
Enter role (admin, guest, user): user
Successfully registered! You can now log in when running the program.
```

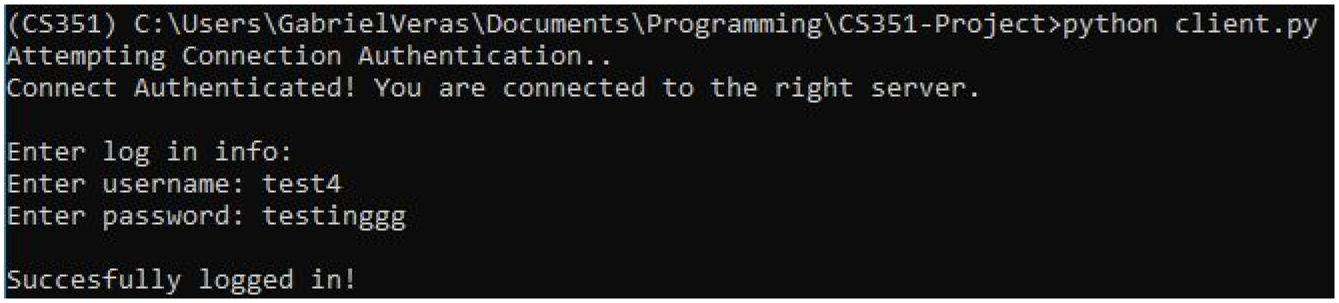
1B. What is stored in the backend



accounts.txt - Notepad

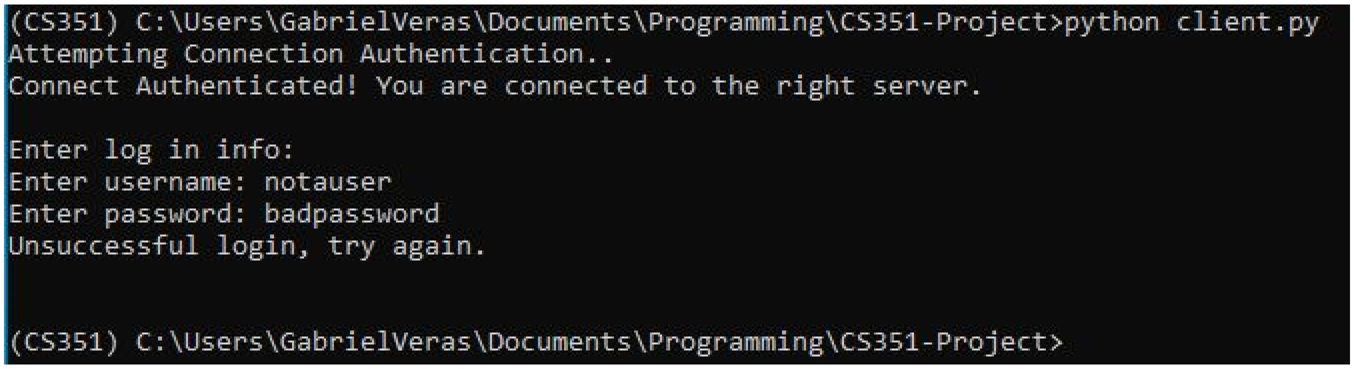
```
guest1,edebe77608538301bc8af157b5c4032e69c95b50f80dab1f1b7ff6068165afeefe2ae0173a6dbf853fe9499b27b1a6e12fc022d7624b8edadd774b9c982aaa3b---f879008db7254c0aa83222d80d647d18,guest
guest2,b245fea8206f4c1e7859df2fe730537d1e25abc6e64fcf878bc229b14ecf32a875909728b8175bcdf5822774a15350678a772292568fc245825c108b79d1da48---d3d23d2453574cd1b3a37e92f58b4dcc,guest
admin1,8c2fad86c51603f460231b56564739ab69e6a287db5cf40531b31700c0fc6caf55993a12add3067f070f954a32d73be14f601c740b9d8c467aa8ee6683b272e5---28969337449247dcb81a90b9ff04c5ed,admin
admin2,0056d6161d6c5a2deb9d467b748f4ffe3313b9b0bda8d5e4852be7f41e1a774942e0c6206b3f5264d63889bd9e82ebf7e20959d1cd0f5953840c226f43938b92---d8d00b0fbeda406783cdb5d4a9e8d3ba,admin
gabe,eca2ae40e2db3f2b913f2027bf2c53e1a69f44ee0130cf0fbd5eac42c5187c8cbbf150a59f15abc7a62d37674a96784bf4e462b0ba2cac3c69ebd4ad2b9e9d33---d14caae1d6384cc397a95212735d1733,user
bob,747edb1af615445229d54ed33ab0b3d70a3270af7a0c8fd012edda25559e278c8dd3ffd93834577bebfdecc2921983afbff9f6fb6c2099d15dc6eeb7f57a5d3a---e4b6d133501c4cf790587f9456fa452e,user
alice,df79304df2eb805e915e6e0916f79dec852d4b9dfd8dfa55cdf93f6f92b3223c6a72740853aa9f8593c3b3e7fc77dde9fa33e4188fb5cee05ff59e47cffa5df6---38c278f6445c40bb8b0ce6e08c5a57d5,user
test,0e0086d2d32e56a78fcb8f320d7fabe89a055fc164921c1547770f9e17c89c002e3024ee32fd01fab9375409669f9489e8bd99735c94d9717bcdb2e9a06fd5c1---bbe4fc8c81cb42e981a518e8629f870d,guest
test2,662c6dc00b787cf9031c70c70d6605f5f9e4c5b6f9c84542c6bf53c6d031b7557f89d87bf2de87b4ba1bfa82e900934215c79f5a3a6549fc22e611253c990374---4a46420da8774f0f96ae1bf9ad317019,user
test3,047df8ba6f462af7d8f77819367453fc8c93aa939d1f44bd1bb98cf00fc6099363a1ce4d93e7d3ad318f7ced686b41ca1ed6f7b39668573ded43df12b5907c36---7ebf5c3a0ad046c2afc45fa6bf2ff75c,user
test4,2af32b9c74619f8dd7da14ca9caa2cddbbff977243a01c030fda66a2f01067705add7971d669547bcc52687ad4ba1d44edbf9cee8e936338094ac72f764aeea5---8c145e20b3bd45f0a47a10c85469540b,user
```

1B. Logging in (Successful) & 2. Connection authentication - connecting to the right server (successful)



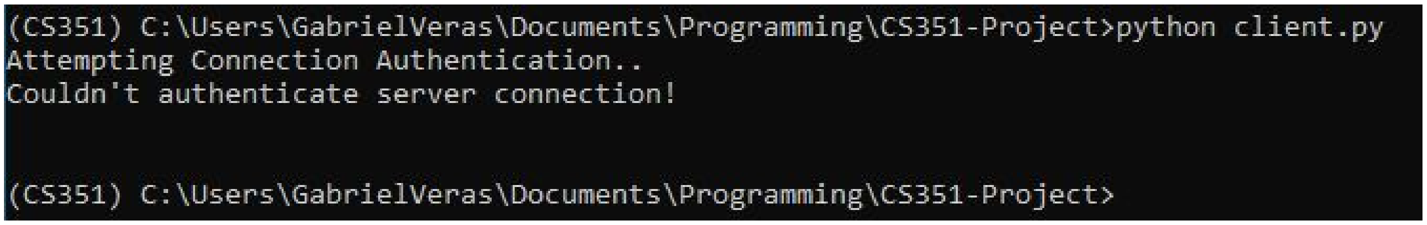
```
(CS351) C:\Users\GabrielVeras\Documents\Programming\CS351-Project>python client.py
Attempting Connection Authentication..
Connect Authenticated! You are connected to the right server.

Enter log in info:
Enter username: test4
Enter password: testinggg

Succesfully logged in!
```

1B. Logging in (Unsuccessful)



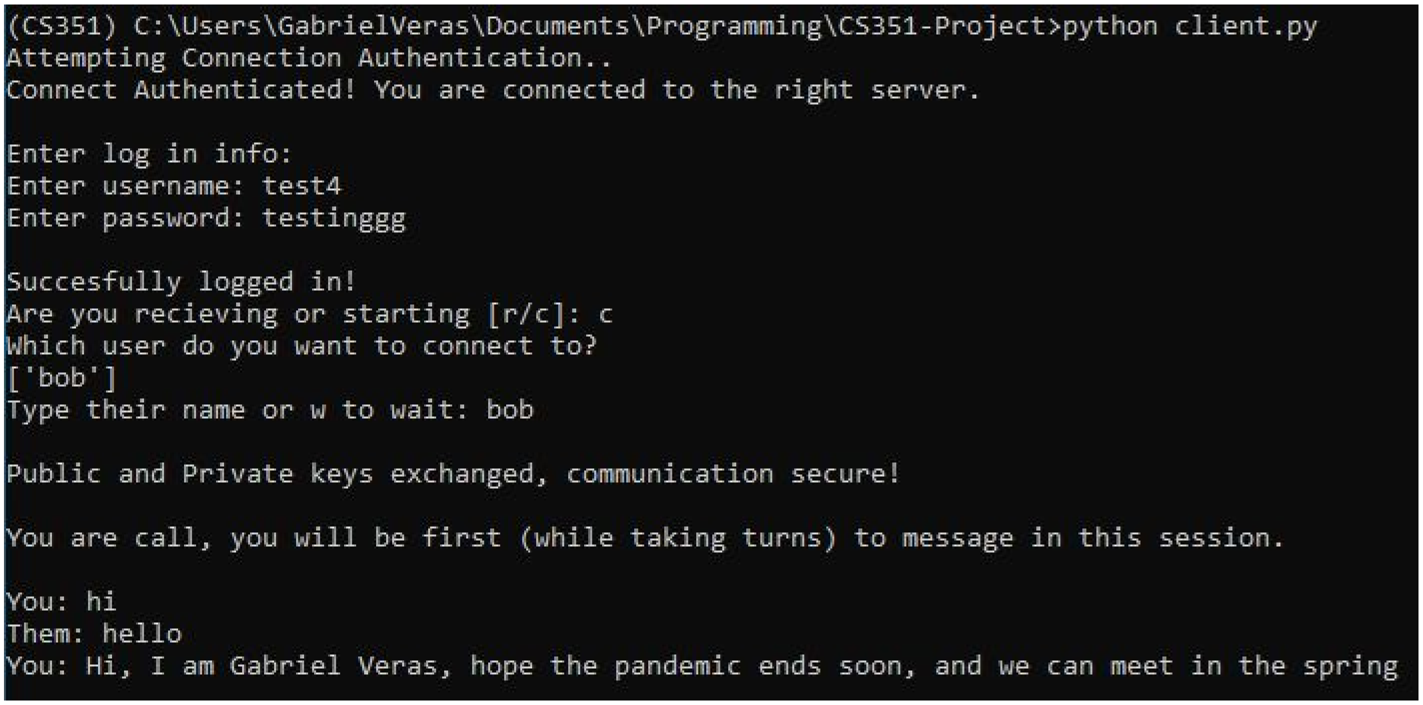
```
(CS351) C:\Users\GabrielVeras\Documents\Programming\CS351-Project>python client.py
Attempting Connection Authentication..
Connect Authenticated! You are connected to the right server.

Enter log in info:
Enter username: notauser
Enter password: badpassword
Unsuccessful login, try again.


(CS351) C:\Users\GabrielVeras\Documents\Programming\CS351-Project>
```

2. Connection authentication - connecting to the right server (unsuccessful)



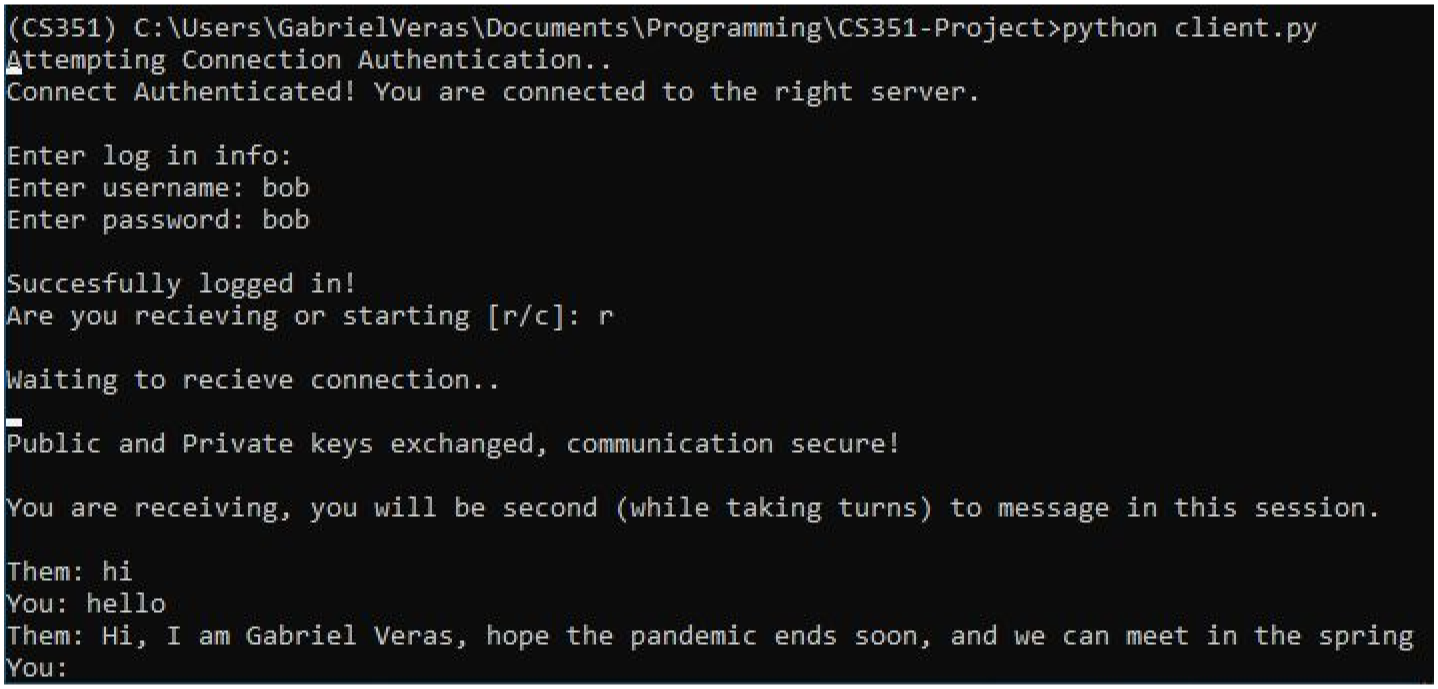
```
(CS351) C:\Users\GabrielVeras\Documents\Programming\CS351-Project>python client.py
Attempting Connection Authentication..
Couldn't authenticate server connection!


(CS351) C:\Users\GabrielVeras\Documents\Programming\CS351-Project>
```

3. Sending the test message (Sender):



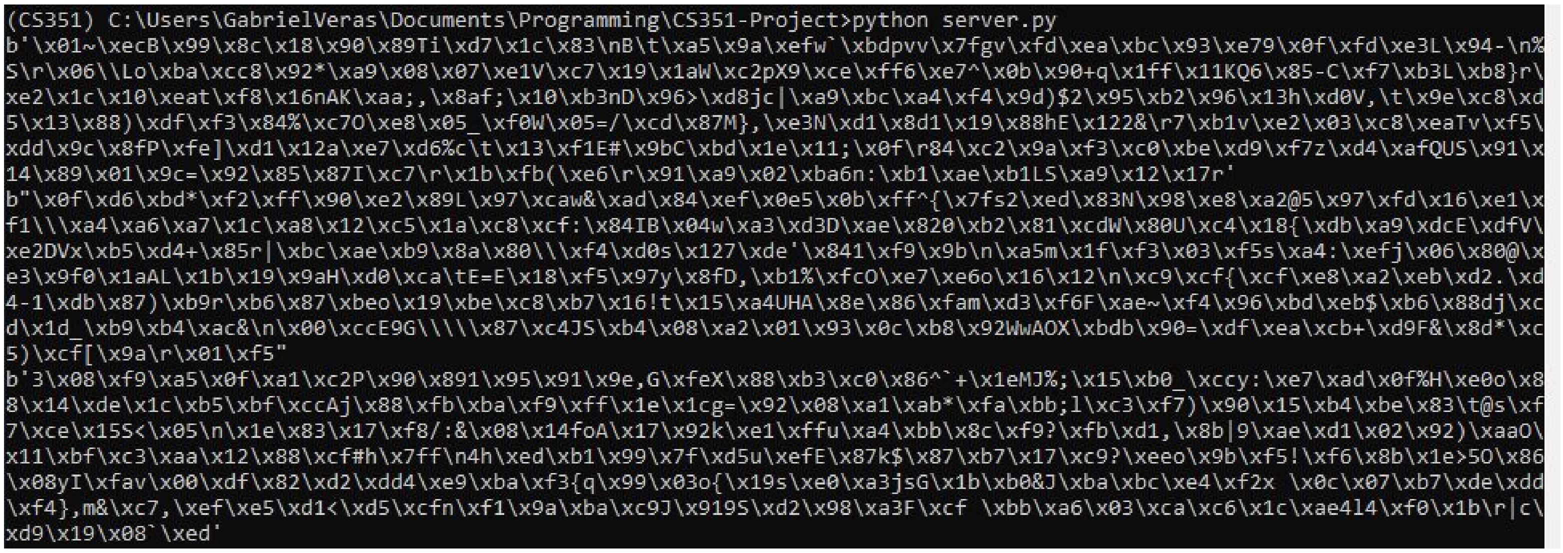
```
(CS351) C:\Users\GabrielVeras\Documents\Programming\CS351-Project>python client.py
Attempting Connection Authentication..
Connect Authenticated! You are connected to the right server.

Enter log in info:
Enter username: test4
Enter password: testinggg

Succesfully logged in!
Are you recieving or starting [r/c]: c
Which user do you want to connect to?
['bob']
Type their name or w to wait: bob

Public and Private keys exchanged, communication secure!

You are call, you will be first (while taking turns) to message in this session.

You: hi
Them: hello
You: Hi, I am Gabriel Veras, hope the pandemic ends soon, and we can meet in the spring
```

3. Sending the test message (Receiver):



```
(CS351) C:\Users\GabrielVeras\Documents\Programming\CS351-Project>python client.py
Attempting Connection Authentication..
Connect Authenticated! You are connected to the right server.

Enter log in info:
Enter username: bob
Enter password: bob

Succesfully logged in!
Are you recieving or starting [r/c]: r

Waiting to recieve connection..

Public and Private keys exchanged, communication secure!

You are receiving, you will be second (while taking turns) to message in this session.

Them: hi
You: hello
Them: Hi, I am Gabriel Veras, hope the pandemic ends soon, and we can meet in the spring
You:
```

3. Sending the test message (Server's View):



```
(CS351) C:\Users\GabrielVeras\Documents\Programming\CS351-Project>python server.py
b'\x01~\xecB\x99\x8c\x18\x90\x89Ti\xd7\x1c\x83\nB\t\xa5\x9a\xefw`\xbdpvv\x7fgv\xfd\xea\xbc\x93\xe79\x0f\xfd\xe3L\x94-\n%
S\r\x06\\Lo\xba\xcc8\x92*\xa9\x08\x07\xe1V\xc7\x19\x1aW\xc2pX9\xce\xff6\xe7^\x0b\x90+q\x1ff\x11KQ6\x85-C\xf7\xb3L\xb8}r\
xe2\x1c\x10\xeat\xf8\x16nAK\xaa;,\x8af;\x10\xb3nD\x96>\xd8jc|\xa9\xbc\xa4\xf4\x9d)$2\x95\xb2\x96\x13h\xd0V,\t\x9e\xc8\xd
5\x13\x88)\xdf\xf3\x84%\xc7O\xe8\x05_\xf0W\x05=/\xcd\x87M},\xe3N\xd1\x8d1\x19\x88hE\x122&\r7\xb1v\xe2\x03\xc8\xeaTv\xf5\
xdd\x9c\x8fP\xfe]\xd1\x12a\xe7\xd6%c\t\x13\xf1E#\x9bC\xbd\x1e\x11;\x0f\r84\xc2\x9a\xf3\xc0\xbe\xd9\xf7z\xd4\xafQUS\x91\x
14\x89\x01\x9c=\x92\x85\x87I\xc7\r\x1b\xfb(\xe6\r\x91\xa9\x02\xba6n:\xb1\xae\xb1LS\xa9\x12\x17r'
b"\x0f\xd6\xbd*\xf2\xff\x90\xe2\x89L\x97\xcaw&\xad\x84\xef\x0e5\x0b\xff^{\x7fs2\xed\x83N\x98\xe8\xa2@5\x97\xfd\x16\xe1\x
f1\\\xa4\xa6\xa7\x1c\xa8\x12\xc5\x1a\xc8\xcf:\x84IB\x04w\xa3\xd3D\xae\x820\xb2\x81\xcdW\x80U\xc4\x18{\xdb\xa9\xdcE\xdfV\
xe2DVx\xb5\xd4+\x85r|\xbc\xae\xb9\x8a\x80\\\xf4\xd0s\x127\xde'\x841\xf9\x9b\n\xa5m\x1f\xf3\x03\xf5s\xa4:\xefj\x06\x80@\x
e3\x9f0\x1aAL\x1b\x19\x9aH\xd0\xca\tE=E\x18\xf5\x97y\x8fD,\xb1%\xfcO\xe7\xe6o\x16\x12\n\xc9\xcf{\xcf\xe8\xa2\xeb\xd2.\xd
4-1\xdb\x87)\xb9r\xb6\x87\xbeo\x19\xbe\xc8\xb7\x16!t\x15\xa4UHA\x8e\x86\xfam\xd3\xf6F\xae~\xf4\x96\xbd\xeb$\xb6\x88dj\xc
d\x1d_\xb9\xb4\xac&\n\x00\xccE9G\\\\\x87\xc4JS\xb4\x08\xa2\x01\x93\x0c\xb8\x92WwAOX\xbdb\x90=\xdf\xea\xcb+\xd9F&\x8d*\xc
5)\xcf[\x9a\r\x01\xf5"
b'3\x08\xf9\xa5\x0f\xa1\xc2P\x90\x891\x95\x91\x9e,G\xfeX\x88\xb3\xc0\x86^`+\x1eMJ%;\x15\xb0_\xccy:\xe7\xad\x0f%H\xe0o\x8
8\x14\xde\x1c\xb5\xbf\xccAj\x88\xfb\xba\xf9\xff\x1e\x1cg=\x92\x08\xa1\xab*\xfa\xbb;l\xc3\xf7)\x90\x15\xb4\xbe\x83\t@s\xf
7\xce\x15S<\x05\n\x1e\x83\x17\xf8/:&\x08\x14foA\x17\x92k\xe1\xffu\xa4\xbb\x8c\xf9?\xfb\xd1,\x8b|9\xae\xd1\x02\x92)\xaaO\
x11\xbf\xc3\xaa\x12\x88\xcf#h\x7ff\n4h\xed\xb1\x99\x7f\xd5u\xefE\x87k$\x87\xb7\x17\xc9?\xeeo\x9b\xf5!\xf6\x8b\x1e>5O\x86
\x08yI\xfav\x00\xdf\x82\xd2\xdd4\xe9\xba\xf3{q\x99\x03o{\x19s\xe0\xa3jsG\x1b\xb0&J\xba\xbc\xe4\xf2x \x0c\x07\xb7\xde\xdd
\xf4},m&\xc7,\xef\xe5\xd1<\xd5\xcfn\xf1\x9a\xba\xc9J\x919S\xd2\x98\xa3F\xcf \xbb\xa6\x03\xca\xc6\x1c\xae4l4\xf0\x1b\r|c\
xd9\x19\x08`\xed'
```