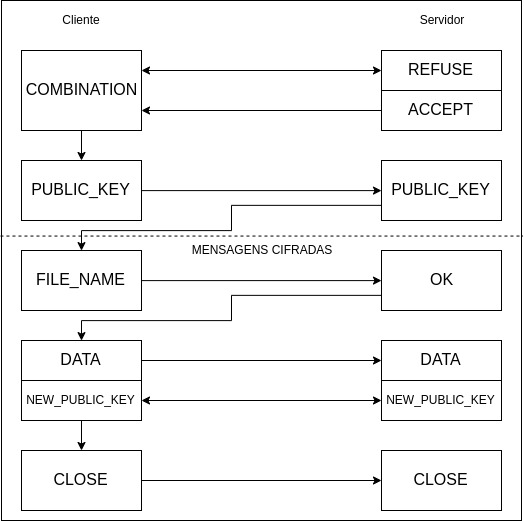
**Mensagens**



Independentemente do estado, caso ocorra algum erro o servidor irá enviar uma mensagem de erro e encerrar a conexão.

{'type': 'ERROR', 'message': 'See server'}

Após ser estabelecida a conexão entre o cliente e o servidor, a primeira mensagem a ser enviada é a da combinação a ser usada, sendo esta enviada pelo cliente.

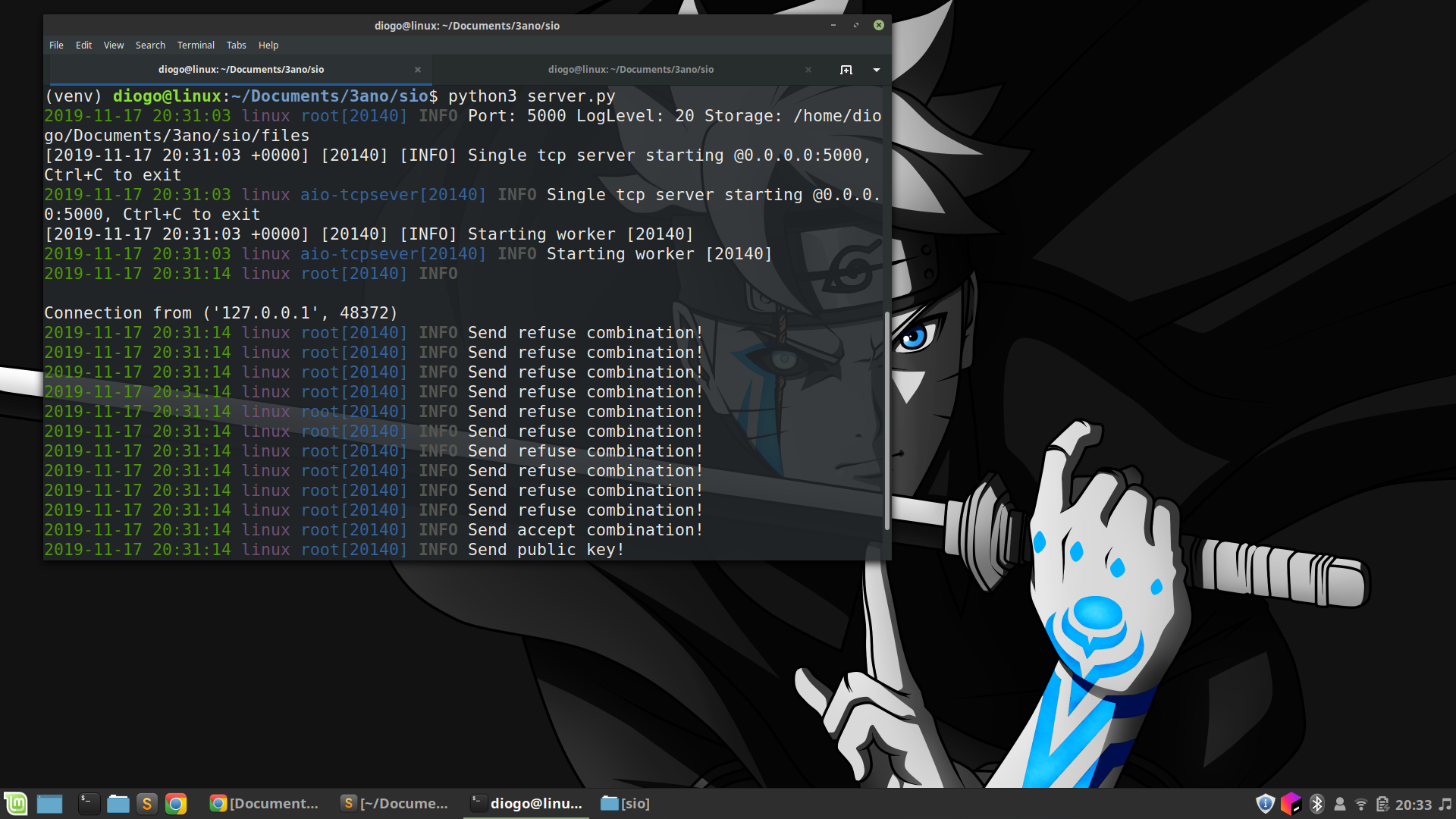
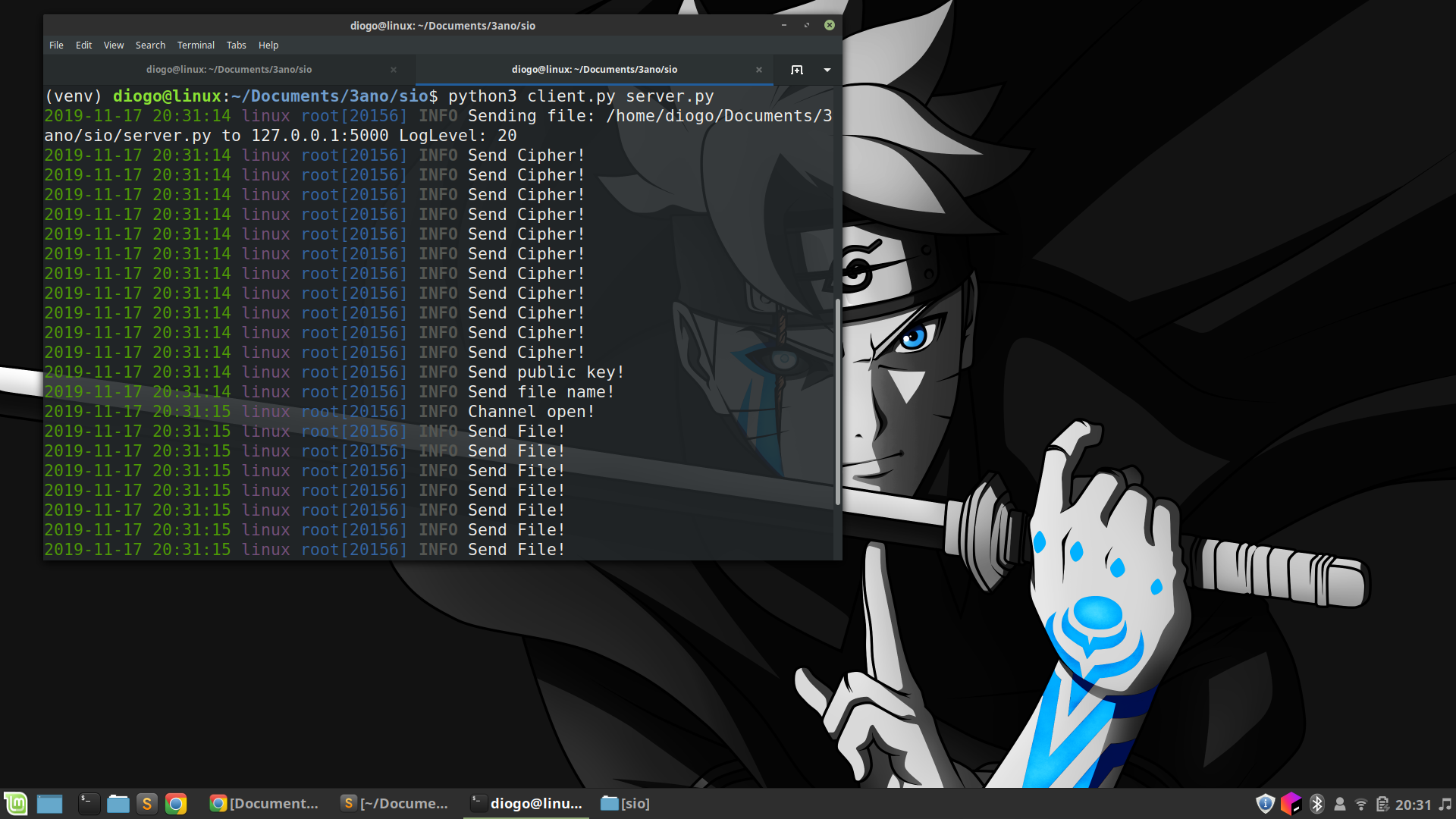
{'type': 'cipher', 'data': COMBINATION}

COMBINATION, exemplo: DH\_AES\_128\_CBC\_SHA512

O servidor pode mandar duas respostas se aceita ou recusa a combinação, ele só rejeita uma combinação caso não a possa usar.

Se recusar o cliente irá enviar outra combinação, existem 3780 combinações, no caso de o servidor recusar todas, o cliente terminará a conexão.

{'type': 'cipher', 'data': 'REFUSE'} ou {'type': 'cipher', 'data': 'ACCEPT'}



Se aceitar o cliente envia a sua chave pública, recebendo como resposta a mesma mensagem, com a chave pública do servidor.

{'type': 'public\_key', 'data': PUBLIC\_KEY}

**A partir de agora todas as mensagens serão cifradas.**

O cliente vai enviar o nome do ficheiro.

{'type': 'OPEN', 'file\_name': FILE\_NAME}

Caso não ocorra nenhum erro o servidor irá enviar a mensagem ‘OK’.

{'type': 'OK'}

A partir deste momento o cliente começa a enviar o ficheiro.

{'type': 'DATA', 'data': DATA}

Durante o envio, caso seja ultrapassado o limite de utilização da chave ele vai enviar uma nova chave pública e receber a do servidor. Depois de concluída a nova troca de chaves, o cliente continuará a enviar o ficheiro.

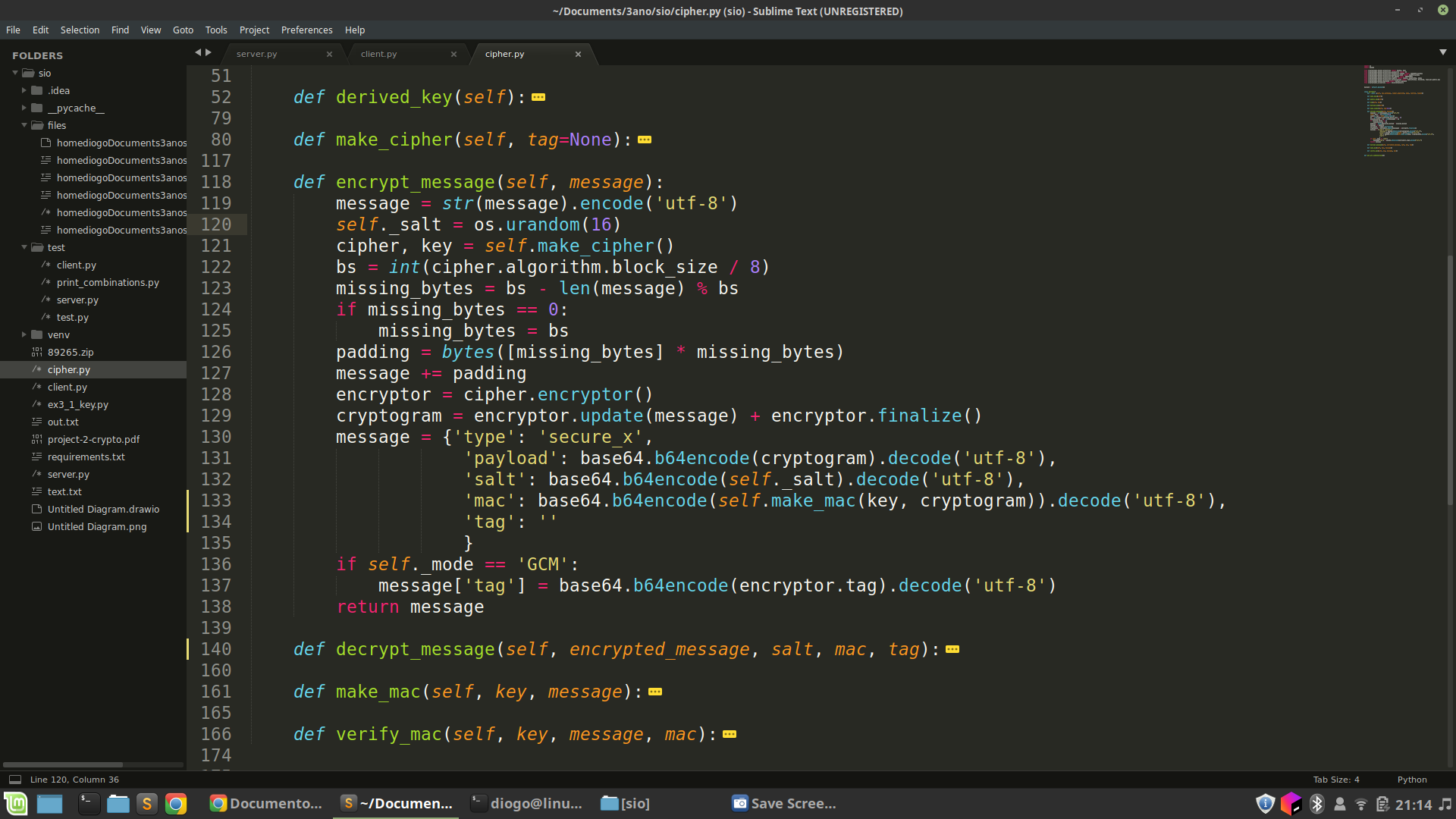
Quando terminar de o enviar, vai fechar a conexão.

{'type': 'CLOSE'}

|  |  |
| --- | --- |
| Servidor | Cliente |
|  |  |

**Cifra das Mensagens**

Para cifrar mensagens foi criada uma nova classe MyCipher, esta é responsável por criar as chaves, encriptar e decriptar as mensagens e gerar e verificar os respetivos mac’s.



O método acima é usado para a encriptação das mensagens, como podemos ver da linha 122 à 127, temos a atenção ao tamanho da mensagem, ou seja, que o tamanho dos blocos para cifrar seja sempre o mesmo.

Ao encriptar a mensagem esta vai ficar com a seguinte estrutura:

message = {

'type': 'secure\_x',

'payload': MENSAGEM\_CIFRADA,

'salt': SALT,

'mac': MAC,

‘tag’: TAG

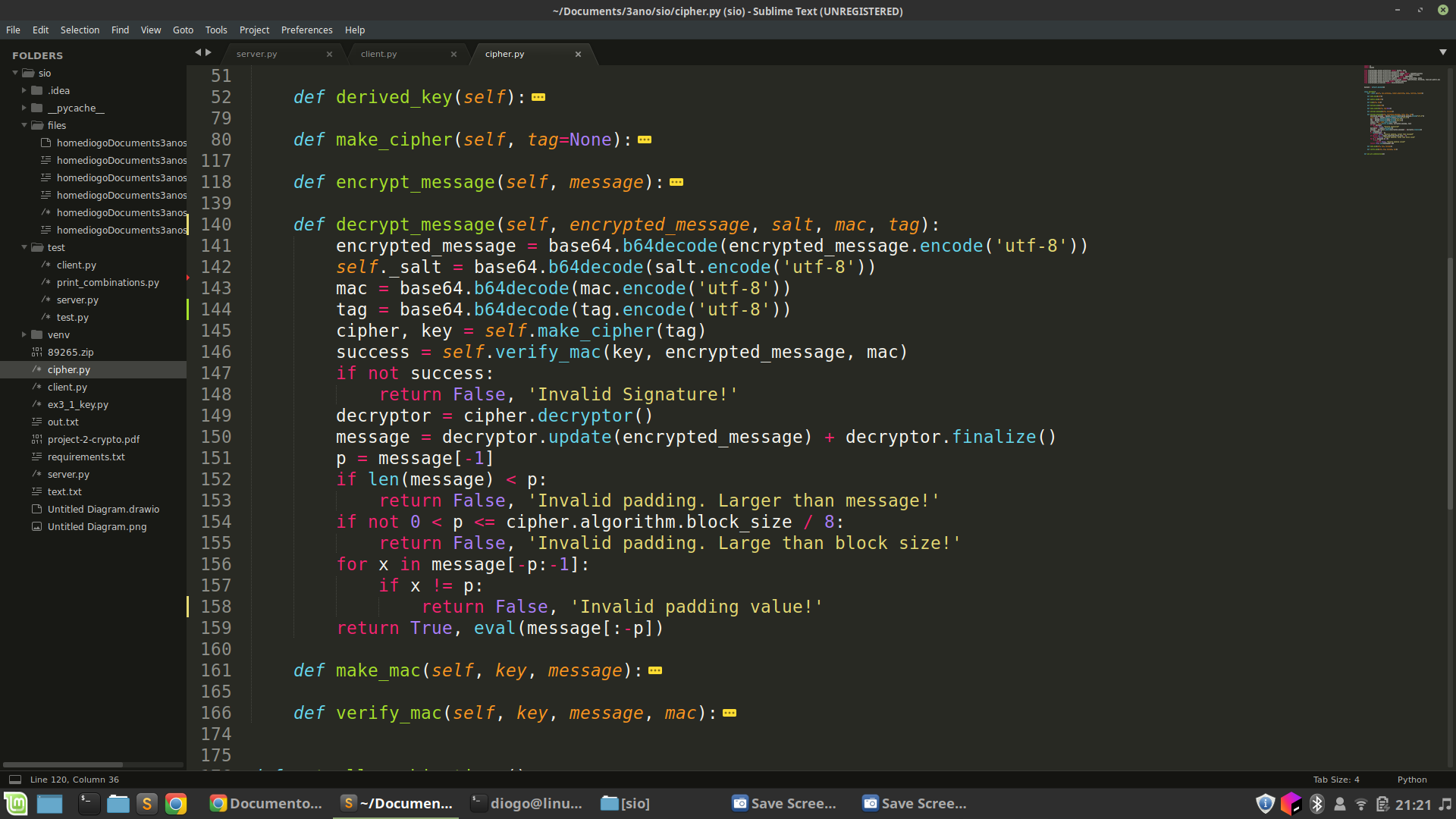
}

MENSAGEM\_CIFRADA são todas mensagens de OPEN, DATA, CLOSE, OK e PUBLIC\_KEY(para novas chaves).

SALT é um valor binário aleatório (os.urandom()).

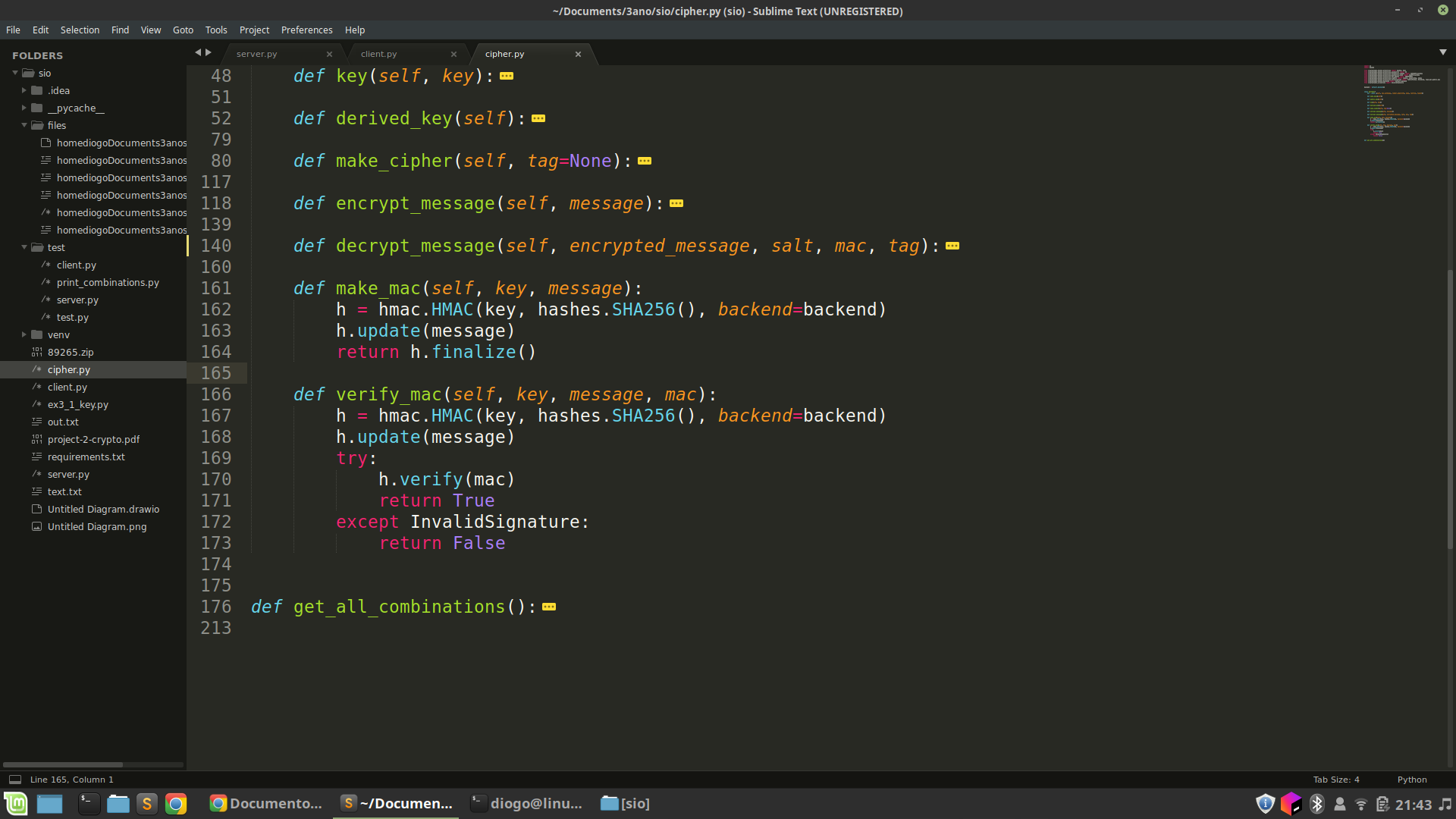
MAC é o autenticador da mensagem.

TAG são os bytes da tag a serem verificados durante a descriptografia, apenas é usado se o mode for GCM (cipher.encryptor().tag).

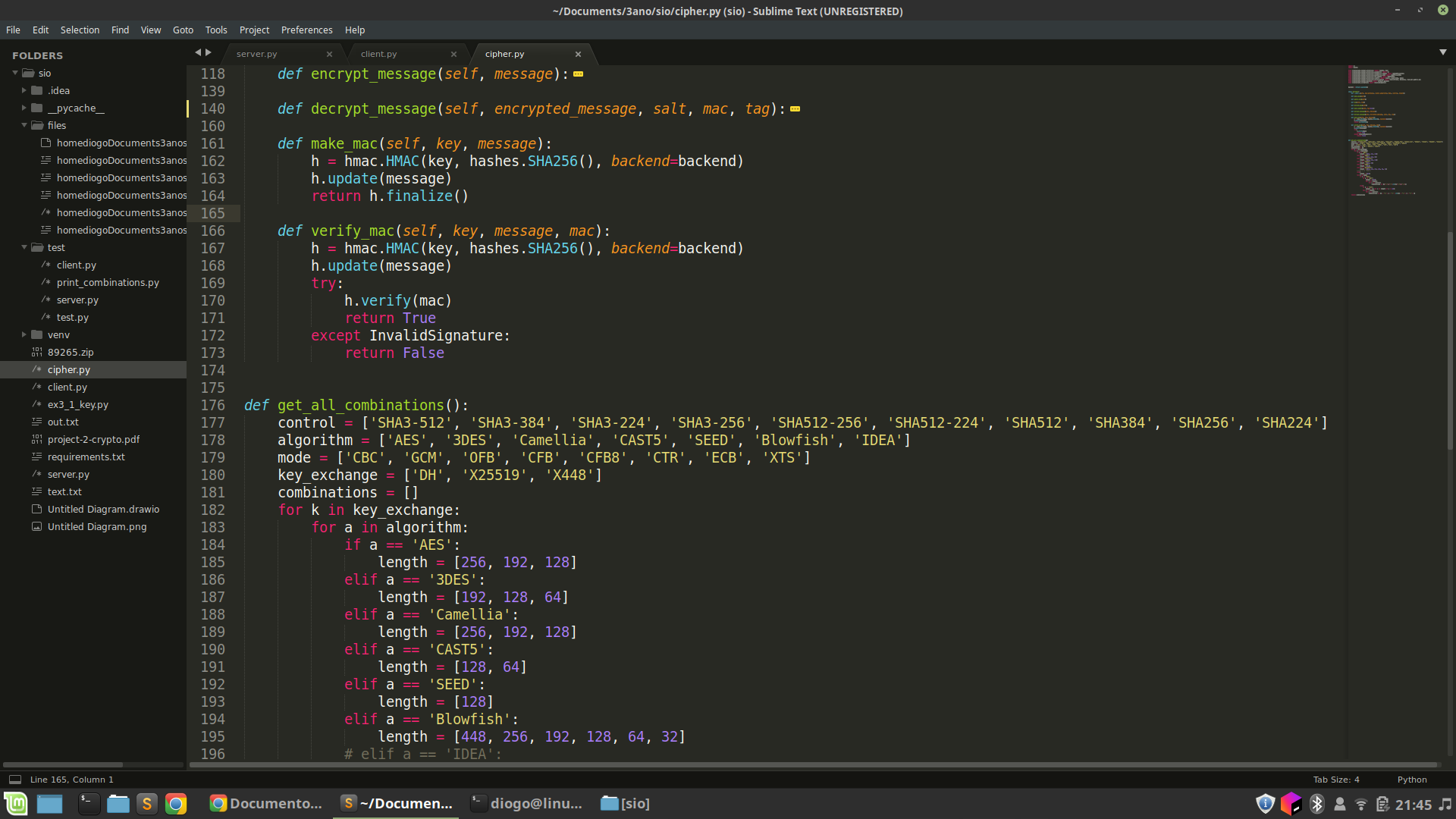


Este método é usado para decriptar as mensagens, mas antes de o fazer ele verifica se o mac corresponde à mensagem recebida e se não houve nenhuma alteração. No caso de não corresponder, o método vai retornar um erro ‘Invalid Signature!’.

À semelhança do que é feito na função de decifra, também verificamos se o tamanho da mensagem é válido, linhas 151 à 158. Caso não haja nenhum problema este vai retornar a mensagem OPEN, DATA, CLOSE, OK ou PUBLIC\_KEY(para novas chaves) decriptada.



Os métodos acima criam e verificam a assinatura de uma mensagem, cada uma tem uma assinatura diferente.



Este método é chamado ao correr tanto o servidor, como o cliente, este retorna o lista de todas as possíveis combinações de algoritmos, hashes, modes (3780 combinações).

