Exercício 1 do TP1

_generate_keys_and*nonces*

Esta função recebe uma chave secreta como entrada que é utilizada para gerar a chave de cifra, a chave de autenticação e o nonce. Isso é feito utilizando o algoritmo Ascon em modo XOF. Os valores gerados são então retornados pela função.

Por fim, a função é chamada com uma chave secreta específica (b'chave_secreta') e os valores resultantes são atribuídos às variáveis key, nonce e associated_data.

In [1]:
```
import asyncio
import ascon
import random
import string

def generate_keys_and_nonces(secret_key):
    prg = ascon.hash(secret_key,variant="Ascon-Xof",hashlength=48)

    cipher_key = prg[:16]  # Tamanho da chave de cifra
    auth_key = prg[16:32]   # Tamanho da chave de autenticação
    nonce = prg[32:48]     # Tamanho do nonce

    return cipher_key, nonce , auth_key

key, nonce, associated_data = generate_keys_and_nonces(b'chave_secreta')
```

_handle*client*

Esta função lida com cada cliente que se conecta ao servidor. Ela recebe uma mensagem do cliente, decifra-a utilizando o algoritmo Ascon, imprime-a e por fim fecha a conexão.

main

A main configura o servidor para aceitar conexões de clientes e inicia o servidor para sempre estar ativo, servindo os clientes.

In [2]:
```
async def handle_client(reader, writer):
    # Lógica de comunicação com o cliente
    message = await reader.read(100)

    # Aqui você pode realizar as operações de cifragem, decifragem, autentic
    # Exemplo: Decifra a mensagem recebida
    decrypted_message = ascon.decrypt(key, nonce, associated_data, message)

    print(f"Received message from client: {decrypted_message}")

    # Não há necessidade de enviar uma resposta ao cliente neste exemplo

    # Fecha a conexão
    writer.close()
```

```
async def main():
    server = await asyncio.start_server(
        handle_client, '127.0.0.1', 8889)

    addr = server.sockets[0].getsockname()
    print(f'Serving on {addr}')

    async with server:
        await server.serve_forever()

asyncio.create_task(main())

# servidor fica á espera de uma msg
```

Out[2]:
```
<Task pending name='Task-5' coro=<main() running at /tmp/ipykernel_2073/348
9250941.py:16>>
Serving on ('127.0.0.1', 8889)
Received message from client: b'msg para cifrar uUWlPWGMsH'
Received message from client: b'msg para cifrar tCZrQExhLE'
```

_send*message*

Esta função estabelece uma conexão com o servidor e envia uma mensagem cifrada. Primeiro, gera uma mensagem aleatória, cifra-a com o algoritmo Ascon e a envia ao servidor. Por fim, fecha a conexão.

In [4]:
```
async def send_message():

    reader, writer = await asyncio.open_connection('127.0.0.1', 8889)

    letters = string.ascii_letters


    message = "msg para cifrar "+''.join(random.choice(letters) for i in ran
    encrypted_message = ascon.encrypt(key, nonce, associated_data, message.e
    writer.write(encrypted_message)
    await writer.drain()

    # Fecha a conexão após enviar a mensagem
    writer.close()
    await writer.wait_closed()

await send_message()

#Cliente envia msg
```