Anonymous API communication

Ex.: ./rep_create_org

Client

(/client folder)

/client.py

rep_create_org()

apiConsumer.send_request(org_data)

/api/api_consumer.py

send_request(org_data)

anonymous_request(org_data)

/utils/client_session_utils.py

anonymous_request(org_data)

exchange_anonymous_keys()

ephemeral_client_public_key = generate_Keypair()

data = {ephemeral_client_public_key}

ECC Pub_K →

Already does integrity check

verify_signature(using repo public key)

generate_shared_secret(using repo ephemeral pub key)

encrypt_anonymous(org_data, shared_secret)

AES + CBC encryption using shared secret

Data
- client_eph_pub_key
- encrypted_data
- IV

So that server knows what encryption key to use (from 'ephemeral_keys' dictionary)

→ Data

decrypt_anonymous(data, shared_secret, IV)

/client.py

show_result(decrypted_response)

Insecure Channel

← Signed ECC Pub_K

Server

(/server folder)

/controllers/organization_controller.py

organizations()

get_ephemeral_server_public_key()

/utils/utils.py

get_ephemeral_server_public_key()

generate_anonymous_signed_shared_secret()

exchange_keys()

ephemeral_server_public_key = generate_Keypair()

generate_shared_secret(from client ephemeral pub key)

data = {ephemeral_server_public_key}

signature = sign(data, server_private_key)

result = {data, signature}

/controllers/organization_controller.py

organizations()

get_shared_secret(from dict, using cli Pub_K)

get_decrypted_request(encrypted_req, shared secret)

return_data = create_organization(decrypted request)

encrypt_anonymous_content(return_data, shared_secret)

data = {encrypted_return_data, IV}

Data
- encrypted_return_data
- IV

← Data