Creating a secure session

Command: ./rep_create_session

"ephemeral key": Used for anonymous content (used once and discarded)
"session key": Used for session content
"cli_session_public_key": ECC key to generate shared session key
"server_session_public_key": ECC key to generate shared session key
"A": Authentication related

Client

(/client folder)

/client.py

rep_create_session()

data = { organization, username }

apiConsumer.exchange_Keys(client private key, data)

Private key corresponding to the public key that was given to the repository when being added to the organization.

/api/api_consumer.py

exchange_Keys(client private key, data)

forwarded to the same function on:

/utils/client_session_utils.py

cli_session_publicKey = generate_Keypair()

data_to_sign = {cli_session_publicKey, data}

A1.1 signature = sign(data_to_sign, client private key)

body = { data_to_sign, signature }

anonymous_request(body)

exchange_anonymous_Keys()

ephemeral_client_public_key = generate_Keypair()

data = { ephemeral_client_public_key }

Insecure Channel

ECC Pub$_K$ →

← Signed ECC Pub$_K$

Server

(/server folder)

/controllers/session_controller.py

sessions()

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 pub Key (ephemeral), ephemeral key)

data = { ephemeral_server_public_key }

A2.1 signature = sign(data, server_private_key)

result = { data, signature }

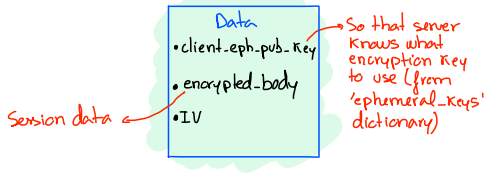
A2.2 verify_signature(using repo public key)

generate_shared_secret(using repo ephemeral pub Key)

ephemeral key

encrypt_anonymous(body, ephemeral_key)

AES + CBC encryption using shared secret (ephemeral)



Data →

/controllers/session_controller.py

sessions()

get_shared_secret (from dict, using cli PUB_ek) — ephemeral key — ephemeral

get_decrypted_request (encrypted_req, ephemeral_key)

return_data = create_session (decrypted request)

body = { data_to_sign, signature }

data_to_sign = {cli_session_public_key, data}

data = { organization, username }

Fetches client public key on that organization, and:

A1.2 verify_signature (data_to_sign, client_pub_key)

exchange_keys()

session_server_public_key = generate_Keypair()

generate_shared_secret (from client session pub key)

→ session key

store on database → create_session (session_data)

result = {
session_id,
username,
organization,
roles,
server_session_public_key,
nonce
}

Session file data (counter will be added on client side)

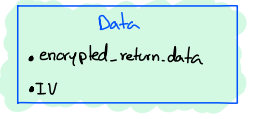
A3.1 signature = sign (result, rep_private_key)

data = { result, signature }

return data

encrypt_anonymous_content (return_data, ephemeral_key)

data = {encrypted_return_data, IV}



← Data

decrypt_anonymous (data, shared_secret, IV)

back to api_consumer.py

A3.1 verify_signature (received_message, rep_pub_key)

session_key = generate_shared_secret (
received_message["server_session_public_key"]
)

/client.py

Finally, store all session data on session file