Challenge Task HS 2018

Alexander van Schie, Elias Brunner, Pascal Bertschi

Requirements

Build a chat application with following architectual details:

- Fully decentralized P2P mechanism
- Communication between peers (one-to-one, group)
- Verification of messages

Project goals

- User registration
- User online status
- User search
- Save contact lists
- Detection of public / private key changes
- Create chat & member invitation
- Message encryption
- Chat list with single / group chats
- Offline messages

User registration

Parameter

Name	Description
username, publicKey, firstName, lastName, walletAddress	User information

Action: PUT

content-key=user,
location-key={username},
value={username, publicKey, firstName, lastName, walletAddress},
signed-with={privateKey}

User search

Parameter

Name	Description
username	Username of searched user

Action: GET

content-key=user,

location-key={username}

Set User online status

Parameter

Name	Description
username	User that the status belongs to
privateKey	Private key of user for signature
status	Status value that the user has

Action: PUT

content-key=onlinestatus,
location-key={username},
value={status},
signed-with={privateKey}
ttl=5mins

Get User Online Status

Parameter

Name	Description
username	User that the online status belongs to

Action: GET

content-key=onlinestatus,
location-key={username}

Save contact list

Parameter

Name	Description
username	User that contact list belongs to
publicKey	Public key of user
privateKey	Private key of user
contactList	List of contacts that get stored

Action: PUT

content-key=contactlist,
location-key={username},
value=contactList=[{username, publicKey, walletAddress, …}, …],
signed-with={privateKey},
encrypted-with={publicKey}

Create chat

Parameter

Name	Description
chatId	Random generated id
name	Name of the new chat (UI)
chatPrivateKey	Random generated key
chatPublicKey	Random generated key

Action: PUT

content-key=chat,
location-key={chatId},
signed-with={chatPrivateKey},
encrypted-with={chatPublicKey},
value={chatId, name, chatPublicKey, chatPrivateKey}

Chat invite

Parameter

Name	Description
chatId, chatName, chatPublicKey, chatPrivateKey	Chat information that user gets invited to
inviteId	Random generated number
username	User that gets the invite
publicKey	Public key of the user that gets the invite

Action: PUT

location-key={username}-invite,
content-key={inviteId},
value={chatId, chatName, chatPublicKey, chatPrivateKey},
encrypted-with={publicKey},
ttl=7days

Save chat list / accept chat invite

Parameter

Name	Description
username	User that chat list belongs to
publicKey	Public key of user for encryption
privateKey	Private key of user for protection
chatList	list of chats that get stored

Action: PUT

content-key=chatlist,
location-key={username},
value={chatList}=[{name, publicKey, privateKey}, …]
encrypted-with={publicKey}
signed-with={privateKey}

Get chat list

Parameter

Name	Description
username	User that chat list belongs to

Action: GET

content-key=chatlist,

location-key={username}

Search invites

Parameter

Name	Description
username	User that the invites belong to

Actions: GET

location-key={username}-invite,

content-key=*

Send message

Parameter

Name	Description
chatId	Chat that message belongs to
messageId	Random generated id
chatPublicKey	Chat public key for encryption
privateKey	User (sender) private key for protection

Action: PUT

location-key={chatId}-messages,
content-key={messageId},
encrypted-with={chatPublicKey},
signed-with={privateKey},
value={sender, message, date}
ttl=7days

Retrieve message

Parameter

Name	Description
chatId	Chat of messages

Action: GET

location-key={chatId}-messages

content-key=*