CryptoBank API Documentation

Data objects

User

RegisterRequest

Field	Type	Description
email	string	6 <= len <= 255, email format
password	string	10 <= len <= 255
confirmedPassword	string	10 <= len <= 255, equal to password
firstName	string	2 <= len <= 50
lastName	string	2 <= len <= 50
phoneNumber	string	3 <= len <= 20, phone format
address	string	5 <= len <= 255

LoginRequest

Field	Type	Description
email	string	6 <= len <= 255, email format
password	string	10 <= len <= 255

LoginResponse

Field	Type	Description
accessToken	string	JWT to be used for authentication
privateKey	string	Base64 encoded private key of the user, new one generated on each login, used for digitally signing transactions.

UserDetails

Field	Type	Description
firstName	string	User’s first name.
lastName	string	User’s last name.
phoneNumber	string	User’s phone number.
address	string	User’s address.

Bank Account

BankAccountCreationResponse

Field	Type	Description
iban	string	IBAN of the newly created bank account.

BankAccountSimpleDetails

Field	Type	Description
creationTime	long	Unix time in milliseconds when the bank account was created.
amount	integer	Amount of money in the bank account.

BankAccountDetails

Field	Type	Description
iban	string	IBAN of the bank account.
creationTime	long	Unix time in milliseconds when the bank account was created.
amount	integer	Amount of money in the bank account.

Transfer

TransferRequest

Field	Type	Description
senderIban	string	IBAN of the sender bank account. Valid IBAN.
receiverIban	string	IBAN of the receiver bank account. Valid IBAN.
amount	integer	Amount of money to be sent. amount > 0

TransferRequestSigned

Field	Type	Description
transferRequest	TransferRequest	The transfer request.
signature	string	Digital Signature of the transferRequest. The bank will verify it with the user's public key to ensure integrity. **For more details, check the Cryptography section**.

TransferDetails

Field	Type	Description
senderIban	string	IBAN of the sender bank account.
receiverIban	string	IBAN of the receiver bank account.
amount	integer	Amount of money to be sent. amount > 0
time	long	Unix time in milliseconds when the transfer started.

isApproved	boolean	Field that represents if the transfer is approved.

Errors

Usually consist of **message: string**, can contain more information based on their type.

Endpoints

User

Register

Path	/api/user/register
Method	POST
Authentication	none
Request data	RegisterRequest
Response data	none
Response codes	**200 OK** - User registered successfully **400 Bad Request** - Validation error / Email already used
Description	Registers a new user to the database.

Login

Path	/api/user/login
Method	POST
Authentication	none
Request data	LoginRequest
Response data	LoginResponse
Response codes	**200 OK** - User logged successfully **400 Bad Request** - Validation error / Invalid credentials
Description	Logs in a user by returning a JWT to be used for authentication and a private key for future transactions. The expiration date of the JWT is set to 1 year. Future iterations should use a refresh token and shorter expiration date.

Get User

Path	/api/user/get
Method	GET
Authentication	Bearer Token with JWT required
Request data	none
Response data	UserDetails
Response codes	**200 OK** **403 Forbidden** - Invalid authentication
Description	Returns a user's details. The user is identified by decrypting the email from the JWT.

Bank Account

Create Bank Account

Path	/api/bank-account/create
Method	POST
Authentication	Bearer Token with JWT required
Request data	none
Response data	BankAccountCreationResponse
Response codes	**200 OK** - Bank account created successfully **403 Forbidden** - Invalid authentication
Description	Creates a new bank account in the database. The starting amount is 0. To change it, make a transfer or alter the database. In a future admin functionality should be added for this. The user is associated with it by decrypting the email from the JWT.

Get Bank Account

Path	/api/bank-account/get/{iban}
Method	GET
Authentication	Bearer Token with JWT required
Request data	Query parameters: Iban - string, max length 34
Response data	BankAccountSimpleDetails
Response codes	**200 OK** **400 Bad Request** - Validation error **403 Forbidden** - Invalid authentication
Description	The user’s bank accounts details.

List Bank Accounts

Path	/api/bank-account/list

Method	GET
Authentication	Bearer Token with JWT required
Request data	none
Response data	List<BankAccountDetails>
Response codes	**200 OK** **403 Forbidden** - Invalid authentication
Description	List containing all the user’s bank accounts details.

Transfer

Transfer Amount

Path	/api/transfer/transfer-amount
Method	POST
Authentication	Bearer Token with JWT required
Request data	TransferRequestSigned
Response data	none
Response codes	**200 OK** - Transfer successful **400 Bad Request** - Invalid transfer data, such invalid iban or amount / Invalid authentication or digital signature **403 Forbidden** - Invalid authentication
Description	Begins a transfer of a certain amount from one of the user's bank accounts to any specified bank account.

List Transfers

Path	/api/transfer/list
Method	GET
Authentication	Bearer Token with JWT required
Request data	none
Response data	List<TransferDetails>
Response codes	**200 OK** **403 Forbidden** - Invalid authentication
Description	List containing all the user's transfers.

Cryptography

Introduction

When the client calls **Login**, alongside the authentication mechanisms the server creates an RSA Key Pair for the user:

1. The public key is stored by the server.
2. The private key is returned to the user in Base64 string encoding. This should be stored by the client.

When a transfer request is called by the client through **Transfer Amount**, alongside the normal authentication through the JSON Web Token, another security measure is added: **Digital Signing**, ensuring that the transfer comes from the client. It works as follows:

Client side

1. A TransferRequest is created, for example:
   {
   “senderIban”: “iban1”,
   “receiverIban”: “iban2”,
   “amount”: 100
   }
2. All whitespaces are removed from the JSON string:
   {“senderIban”:“iban1”,“receiverIban”:“iban2”,“amount”:100}
3. The string is **hashed** using **SHA-256**, with the result being a **lowercase** Base16 string:
   cb266d4ea0b6cff3edc5812532f00b9d45396003326bd061566cc4c75ceb2f8f
4. The hash string is **encrypted** using the user’s **RSA Private Key** stored from the last **Login** call, resulting in a Base64 encoded string called the **signature**.
5. The original TransferRequest is sent alongside the **signature** in a TransferRequestSigned object:
   {
   “transferRequest”: {
   “senderIban”: “iban1”,
   “receiverIban”: “iban2”,
   “amount”: 100
   },
   “signature”: “Base64 Encoded signature”
   }

Server side

1. After authenticating the user, the server takes the **transferRequest** part of the TransferRequestSigned object received from the client and follows the same steps as the client to get the **SHA-256 hash**.
2. The server takes the **signature** part of the TransferRequestSigned object received from the client and with stored user's **RSA Public Key**, it **decrypts** the signature to get the user's hash.
3. The client's **decrypted hash** is compared against the **server's computed hash**. If the two are the **same**, this means that the client who sent the transfer owns the private key generated alongside the server's public key, so the transfer is digitally signed and can be trusted.