Insurance & Banking: BlingBank

Network and Computer Security

Diogo Pereira 110996 | João Santos 110947 | João Matos 110846

Secure Document Format

Secure Document Format

- **The protected document must ensure the authenticity and confidentiality of the account data.**
- Assumption: user and the service share a secret key.


```
{
  "account": {
    "accountHolder": ["Alice"],
    "balance": 872.22,
    "currency": "EUR",
    "movements": [
      {
        "date": "09/11/2023",
        "value": 1000.00,
        "description": "Salary"
      },
      {
        "date": "15/11/2023",
        "value": -77.78,
        "description": "Electricity bill"
      },
      {
        "date": "22/11/2023",
        "value": -50.00,
        "description": "ATM Withdrawal"
      }
    ]
  }
}
```

Secure Document Flow

protect (twoLayer == true)

plain_text_json → **Server** → encrypt with the **account_symmetric_key** (first layer) →

JSON
accountHolder: “alice”
balance: SqdFYB5wT
movements: {
  value: CDT8Q4K5jDM==
  date: ZGS4a2oMtzO31=
  description: lOskTiyTHWRTJ
}

Secure Document Flow

protect (twoLayer == true)

plain_text_json → **Server** → encrypt with the **account_symmetric_key** (first layer)

JSON
accountHolder: “alice”
balance: SqdFYB5wT
movements: {
  value: CDT8Q4K5jDM==
  date: ZGS4a2oMtzO31=
  description: lOskTiyTHWRTJ
}

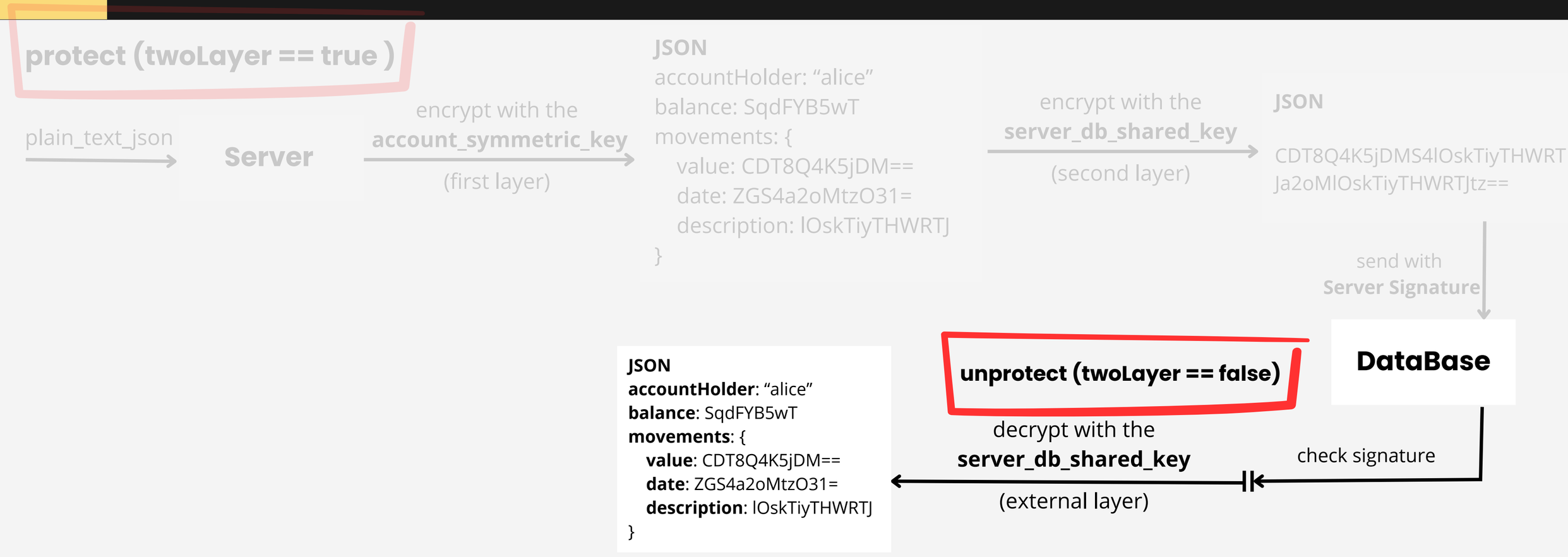
→ encrypt with the **server_db_shared_key** (second layer)

JSON

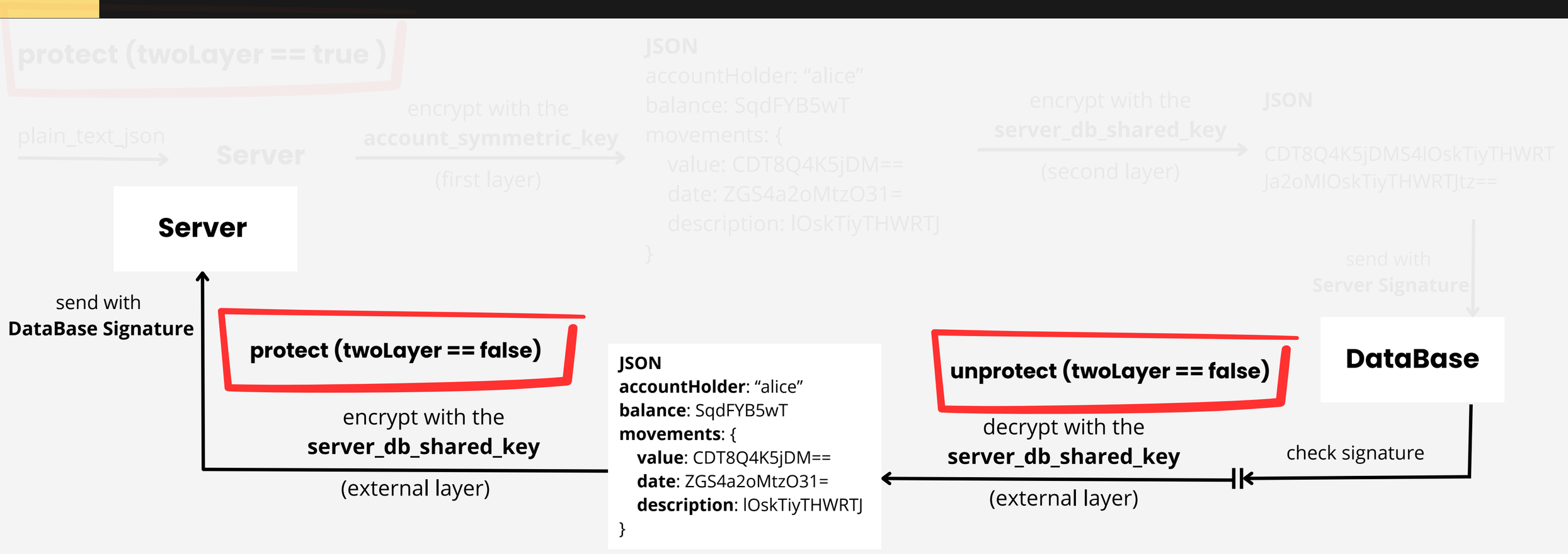
CDT8Q4K5jDMS4lOskTiyTHWRT
Ja2oMlOskTiyTHWRTJtz==

→ send with **Server Signature** → **DataBase**

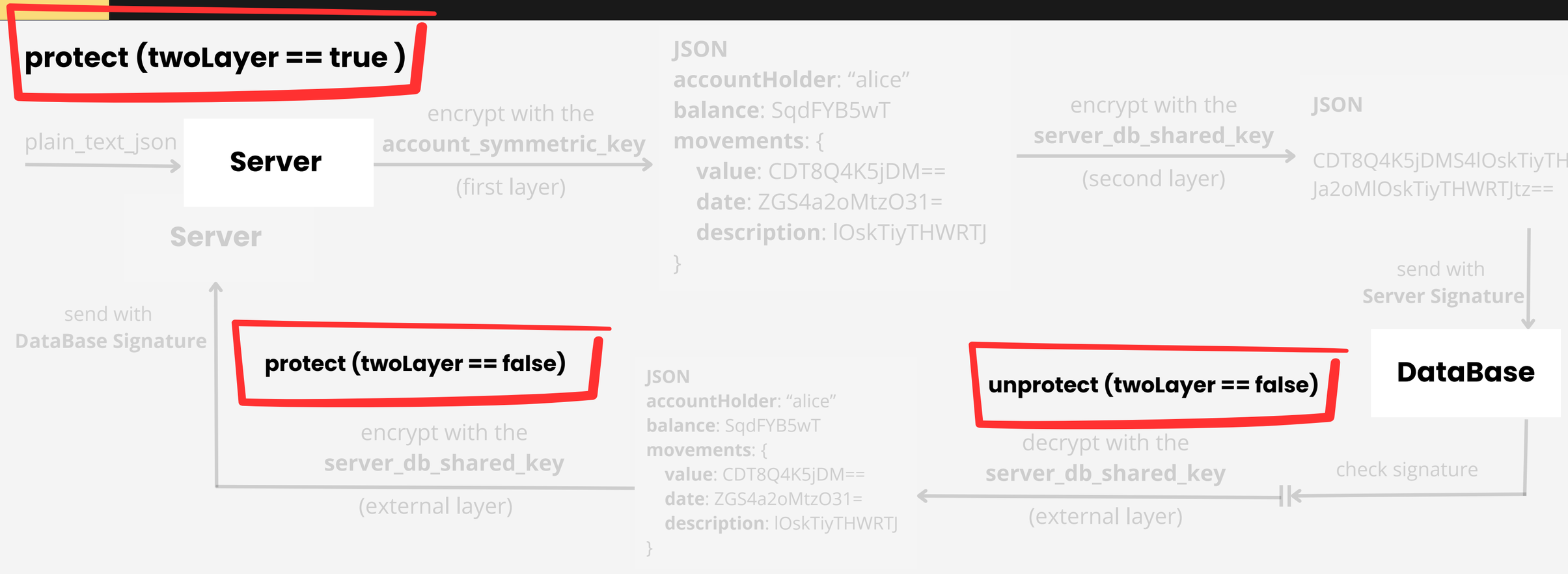
Secure Document Flow

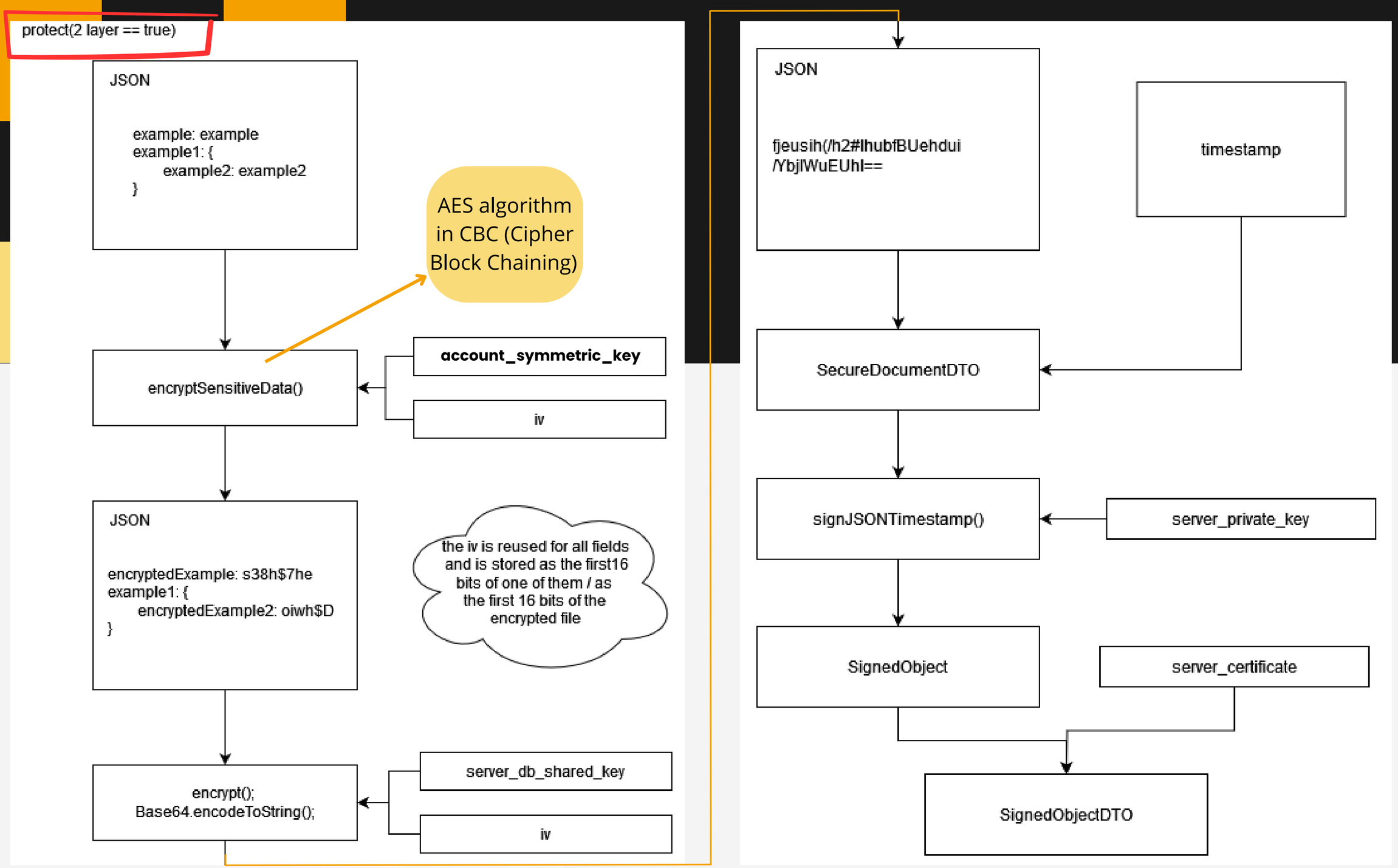


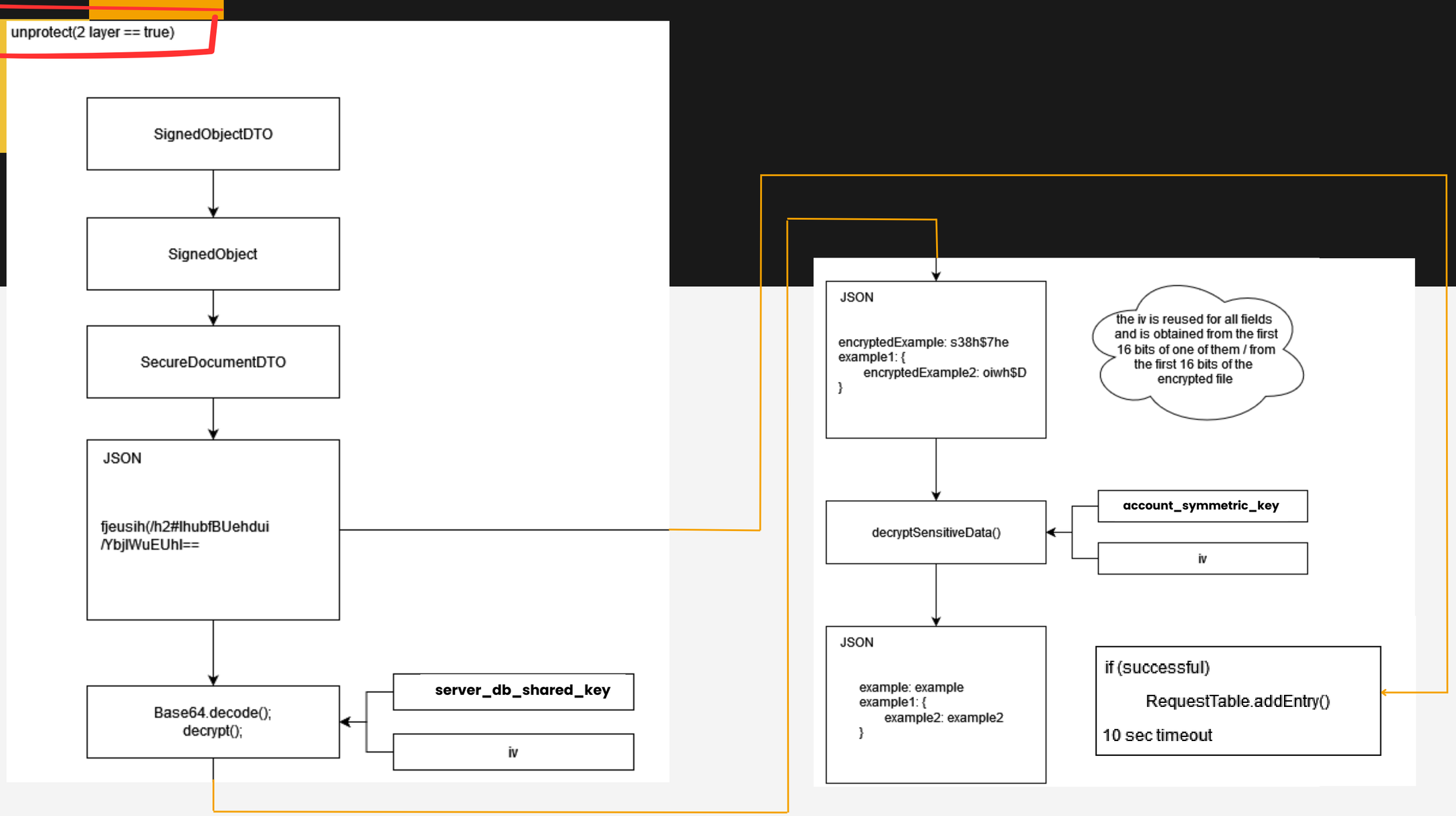
Secure Document Flow



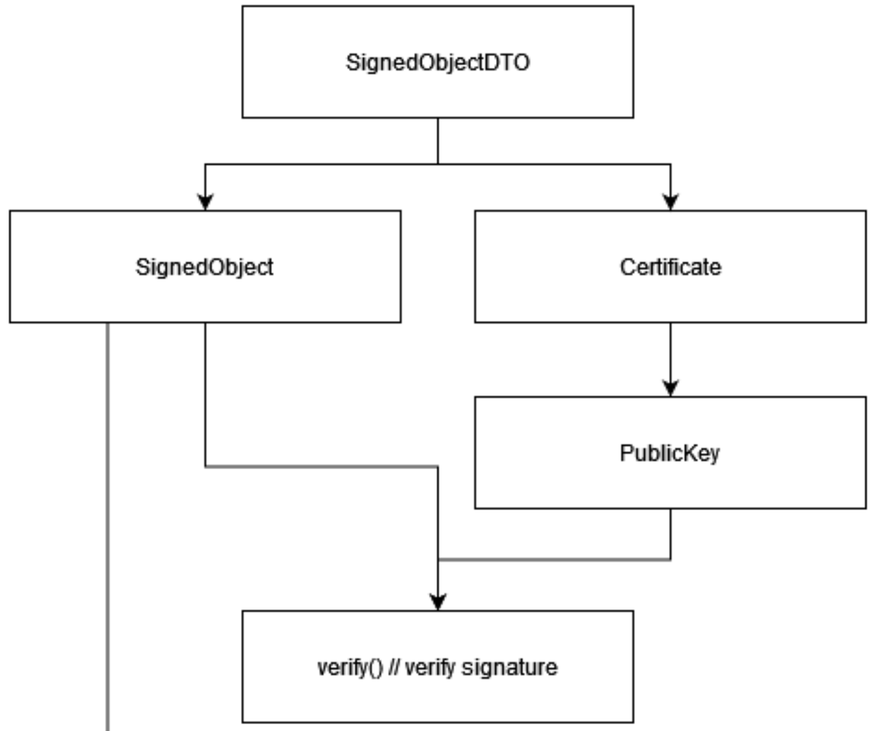
Secure Document Flow



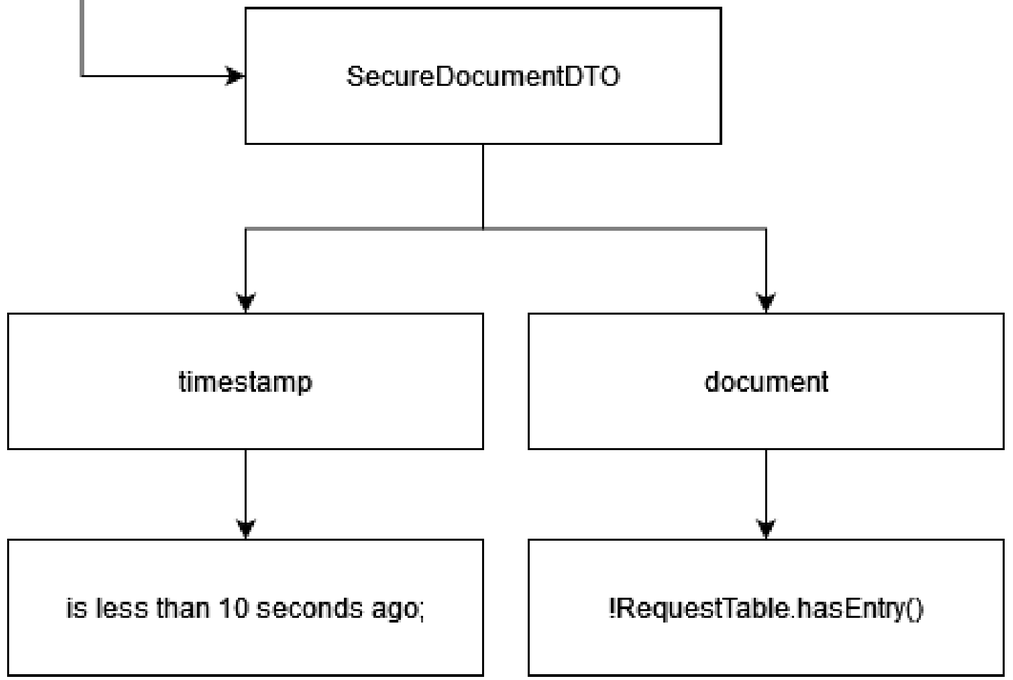

protect(2 layer == true)
JSON
example: example
example1: {
example2: example2
}
AES algorithm in CBC (Cipher Block Chaining)
encryptSensitiveData()
account_symmetric_key
iv
JSON
encryptedExample: s38h$7he
example1: {
encryptedExample2: oiwh$D
}
the iv is reused for all fields and is stored as the first16 bits of one of them / as the first 16 bits of the encrypted file
encrypt();
Base64.encodeToString();
server_db_shared_key
iv
JSON
fjeusih(/h2#IhubfBUehdui
/YbjlWuEUhI==
timestamp
SecureDocumentDTO
signJSONTimestamp()
server_private_key
SignedObject
server_certificate
SignedObjectDTO


unprotect(2 layer == true)
SignedObjectDTO
SignedObject
SecureDocumentDTO
JSON
fjeusih(/h2#IhubfBUehdui
/YbjlWuEUhl==
Base64.decode();
decrypt();
server_db_shared_key
iv
JSON
encryptedExample: s38h$7he
example1: {
encryptedExample2: oiwh$D
}
the iv is reused for all fields and is obtained from the first 16 bits of one of them / from the first 16 bits of the encrypted file
decryptSensitiveData()
account_symmetric_key
iv
JSON
example: example
example1: {
example2: example2
}
if (successful)
RequestTable.addEntry()
10 sec timeout

check



if(successful) continue; else return false



if(both) return true; else return false

Secure Document in DataBase

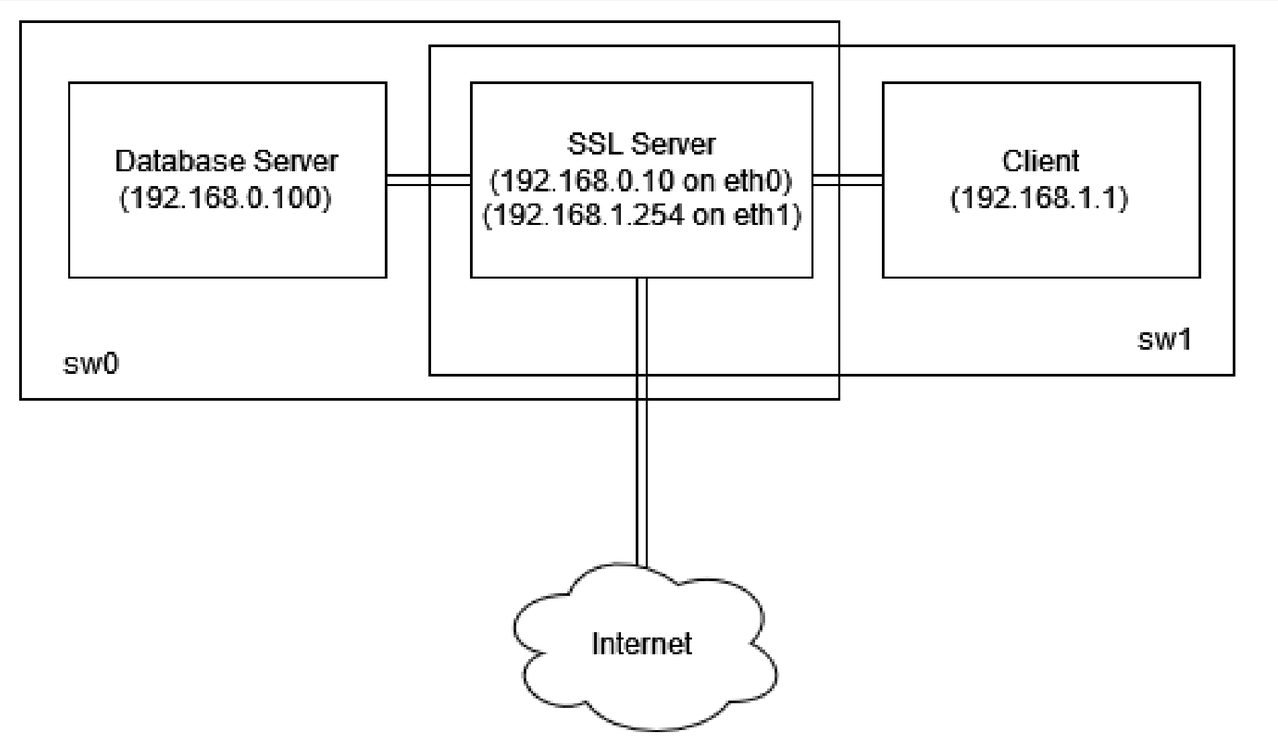

```
_id: ObjectId('65948dd3b8aef664e0c3d1d0')
▾ accountHolder: Array (1)
    0: "alice"
▾ movements: Array (3)
  ▾ 0: Object
      encryptedValue: "CDT8Q4K5jDMZxeH1cx/nXw=="
      encryptedDate: "ZGS4a2oMtzO31+9zJvPMSOSxZy5jhKZwEOuSeSACVBE="
      encryptedDescription: "lOskTiyTHWRTJPUGrexWDw=="
  ▾ 1: Object
      encryptedValue: "a1drMakVgiIdaE13kiSuog=="
      encryptedDate: "0imuQaM6R1TTJi5iN/TdelXG4kzeINufcT7ND7++zvs="
      encryptedDescription: "8mghYqZbJYsEC2aF0iGp5ipwYjkjGb7pabj1WxsMhWY="
  ▾ 2: Object
      encryptedValue: "RVnpP4bV8pf0cZtoqa6S1Q=="
      encryptedDate: "ULbR/Z3ZXODjSxEf+pJuJ/FScwzBZrawn7ymxflh9Ts="
      encryptedDescription: "YhTZoVGexgAPVs0z2ee0SA=="
  encryptedBalance: "O9zsqXoN/SqdFYB5wToSd9RgtLDIRlRfES3TGTJAaPg="
  encryptedCurrency: "Y/n5H1tcgYUCBlE299Cv1w=="
```

Built Infrastructure

Infrastructure

- **The infrastructure is composed of the Database Server, SSL Server and the external clients**
  - Database Server stores an accesses data on MongoDB
  - SSL Server handles client requests

Infrastructure



Key Distribution

Key Distribution

- RSA keys in Server and DataBase, Secret Key shared between both, Certificates changed by the admin
- Secret Key in Server for each Account and Payment Account
- Assumed Secret Key between Server and Client
- Clients obtain Server Certificate in CA
- Automatic generation of RSA keys for each client in first connection
- Client sends Certificate to Server using HMAC, calculated with the Secret Key shared with Server

Secure Channels

Secure Channels

- **SSL sockets were used for Client -> SSL Server and SSL Server -> DataBase Server communications**
  - Truststores are used to store certificates of known entities
  - Keys used during communication are stored on Keystores

Secure Channels

- **For communications between clients and the SSL Server, a new cryptographic library was created: Secure Message Lib**
  - Messages are encrypted using the shared key, then signed with the private key
  - In decryption, the signature is validated using the public key, then, the content is recovered using the shared key

Security Challenge

Security Challenge

- **Guarantee confidentiality, authenticity, and non-repudiation**
- **Freshness measures using timestamps and Request Table**

Main Results and Conclusions

Main Results and Conclusions

- **Robust Cryptography**
  - ensures confidentiality, integrity, authenticity and non-repudiation of transmitted data.
- **Server-Database Communication Security**
  - Bilateral authentication with certificates for Server and Database
- **Client-Server Communication Security**
  - HMAC calculation for Client Certificate enhances integrity.
  - Secure generation and transmission of asymmetric keys for new device connections.
- **Effective prevention of replay attacks**

Live Demonstration