YELLING SECRETS INTO A CROWD: PRIVATE DOCUMENT SHARING ON A PUBLIC BLOCKCHAIN

Ajay Shridhar Joshi, Benita Kathleen Britto, Michael Yanagisawa

PROBLEM STATEMENT

How can users share private documents securely over a blockchain?
Example: sharing medical records

DOCUMENT SHARING

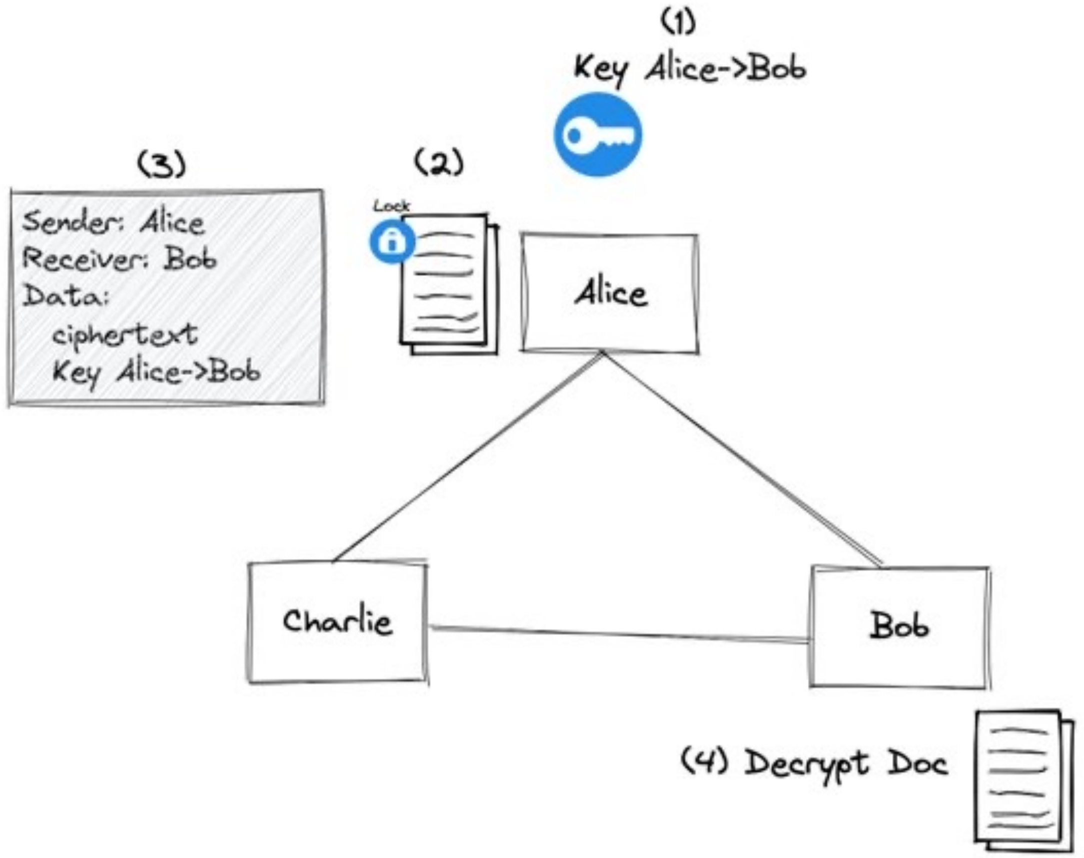
MAINTAIN DATA PRIVACY AND ANONYMITY

INSPIRED BY MONERO

DOCUMENT SHARING: NAÏVE APPROACH

Idea:
- Add text and encryption to project 2b (a simple consortium blockchain)

Steps:
- Establish trust between Alice and Bob with Diffie-Hellman and generate shared secret
- Encrypt document data in the transaction using shared secret



DOCUMENT SHARING: OUR APPROACH

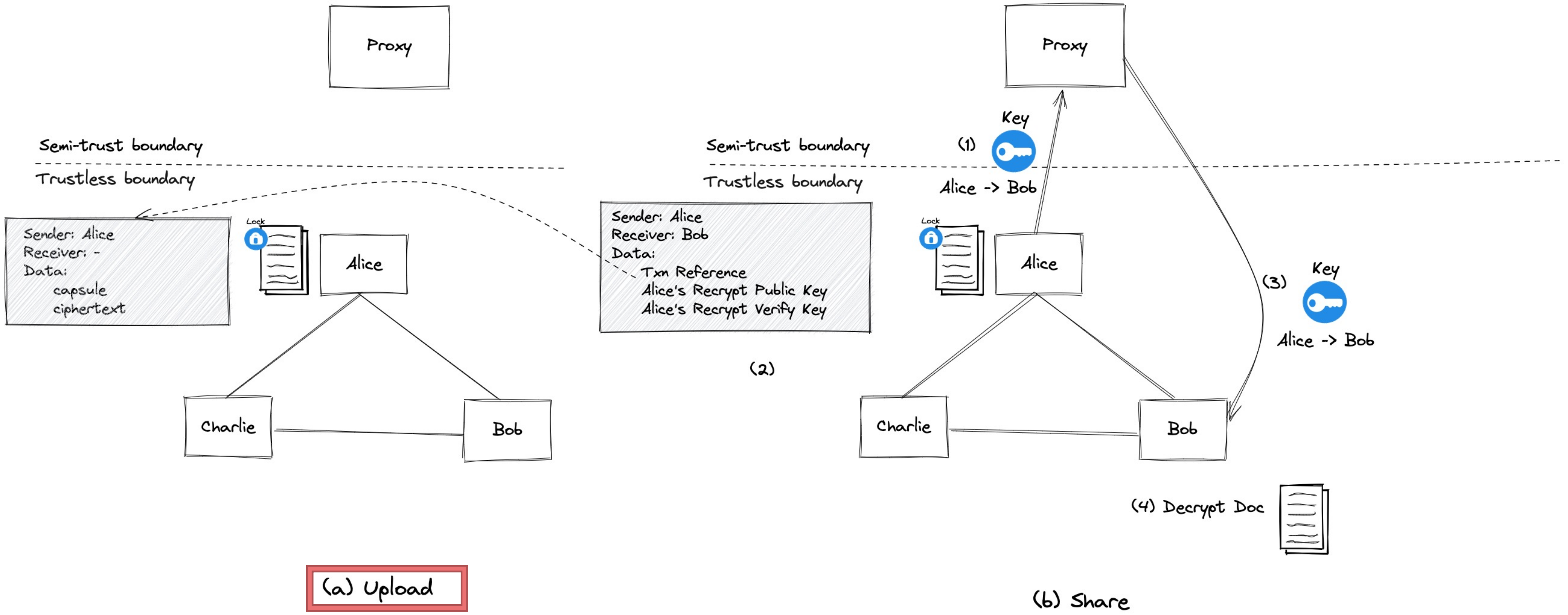
Problem #1: Diffie-Hellman only establishes trust between 2 parties. What if Alice wants to share a document with Bob, Charlie and Don but share only single copy of a file?

Solution: Proxy re-encryption

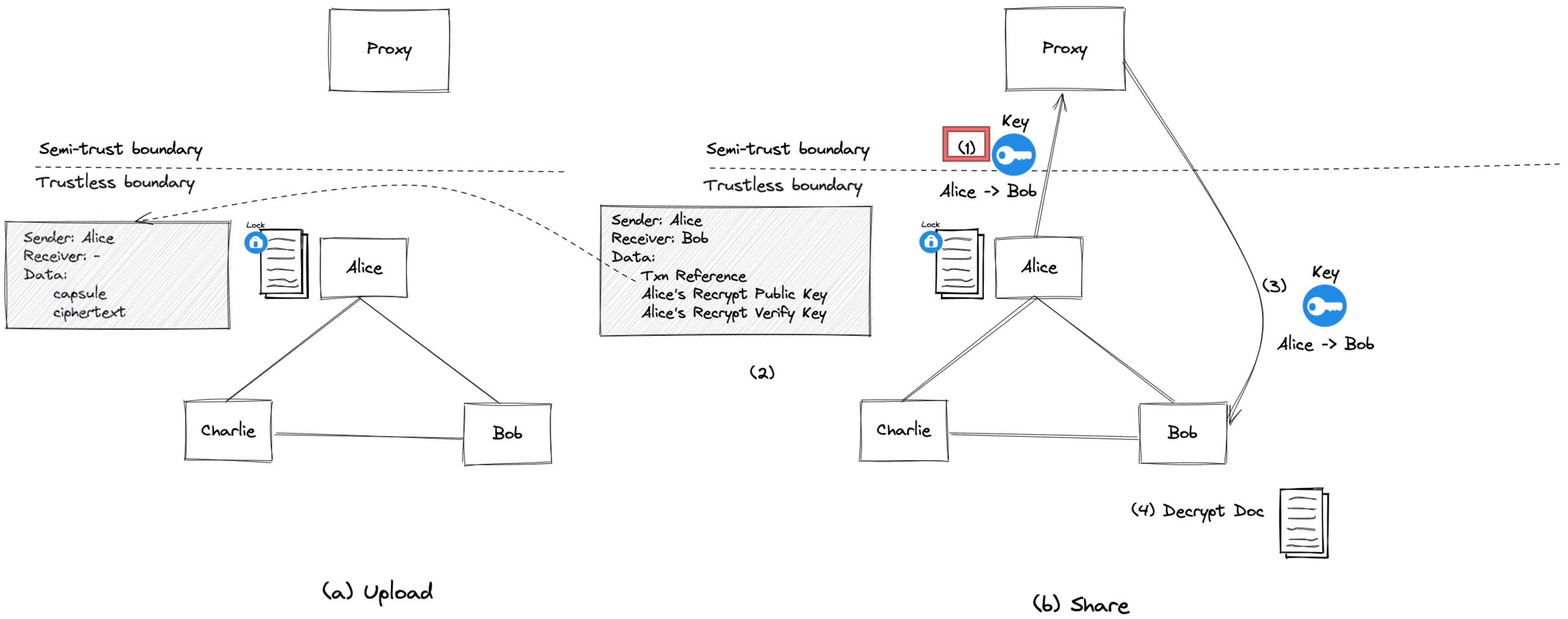
Problem #2: Blockchain ledger leaks metadata, so is not anonymous

Solution: Establish sender and receiver **anonymity** using techniques from Monero

PROXY RE-ENCRYPTION



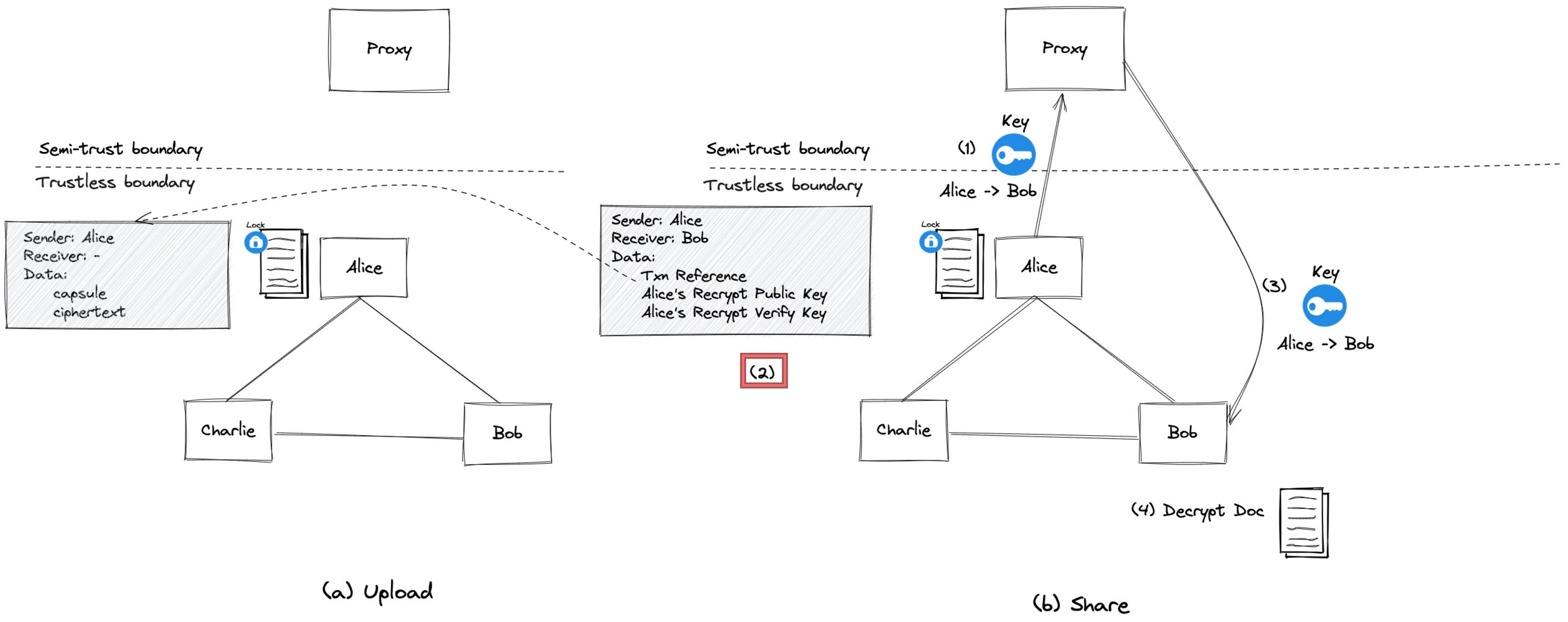
PROXY RE-ENCRYPTION



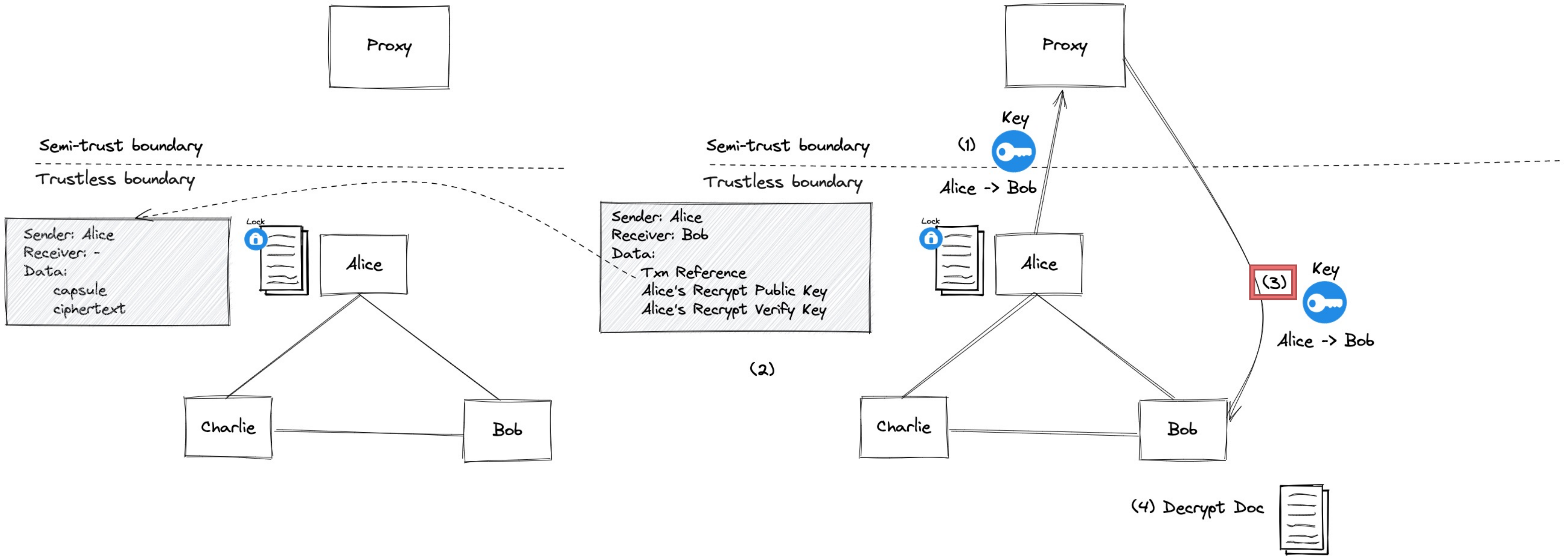
Sender: Alice
Receiver: -
Data:
capsule
ciphertext

Sender: Alice
Receiver: Bob
Data:
Txn Reference
Alice's Recrypt Public Key
Alice's Recrypt Verify Key

PROXY RE-ENCRYPTION



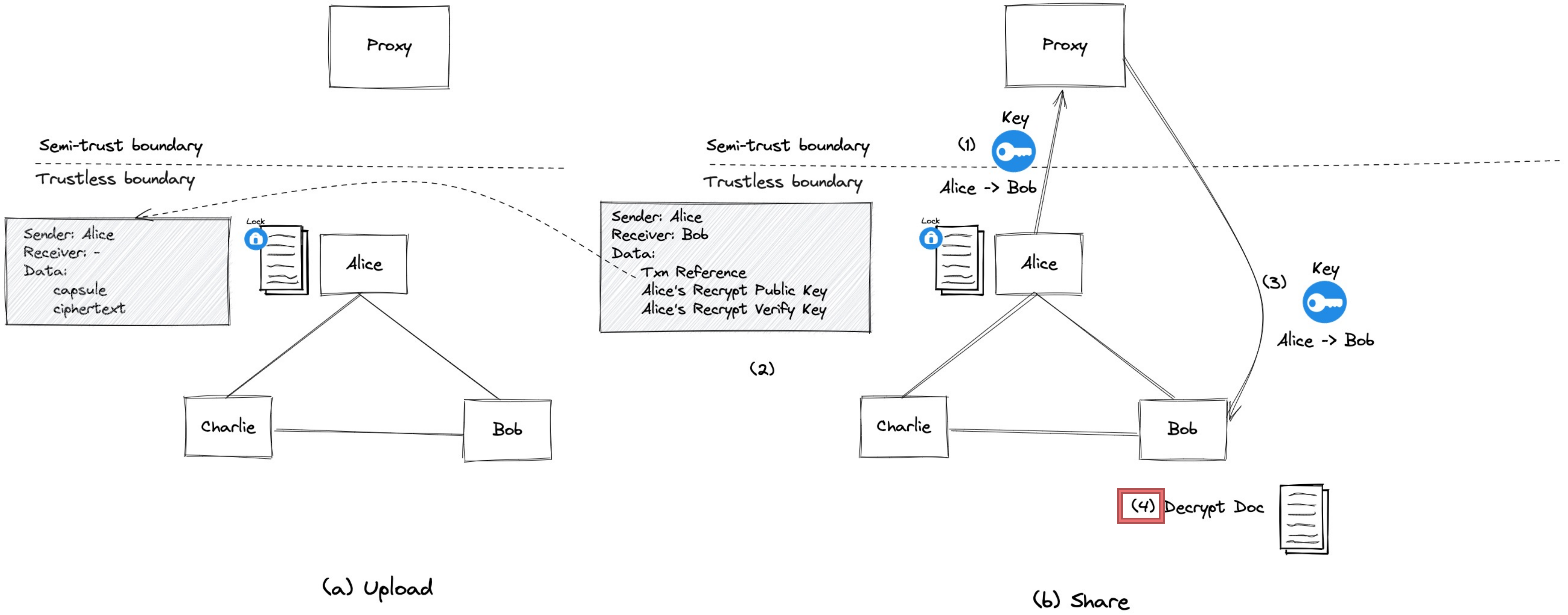
PROXY RE-ENCRYPTION



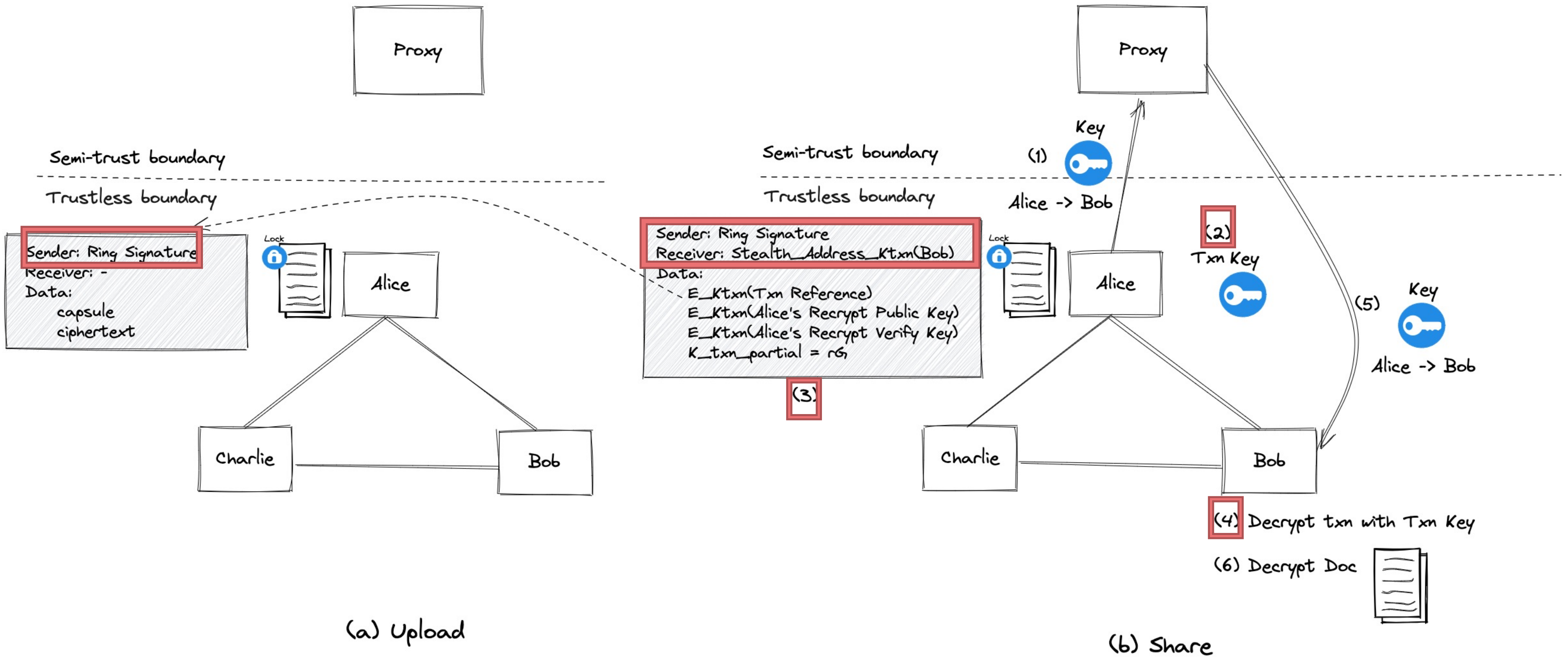
(a) Upload

(b) Share

PROXY RE-ENCRYPTION

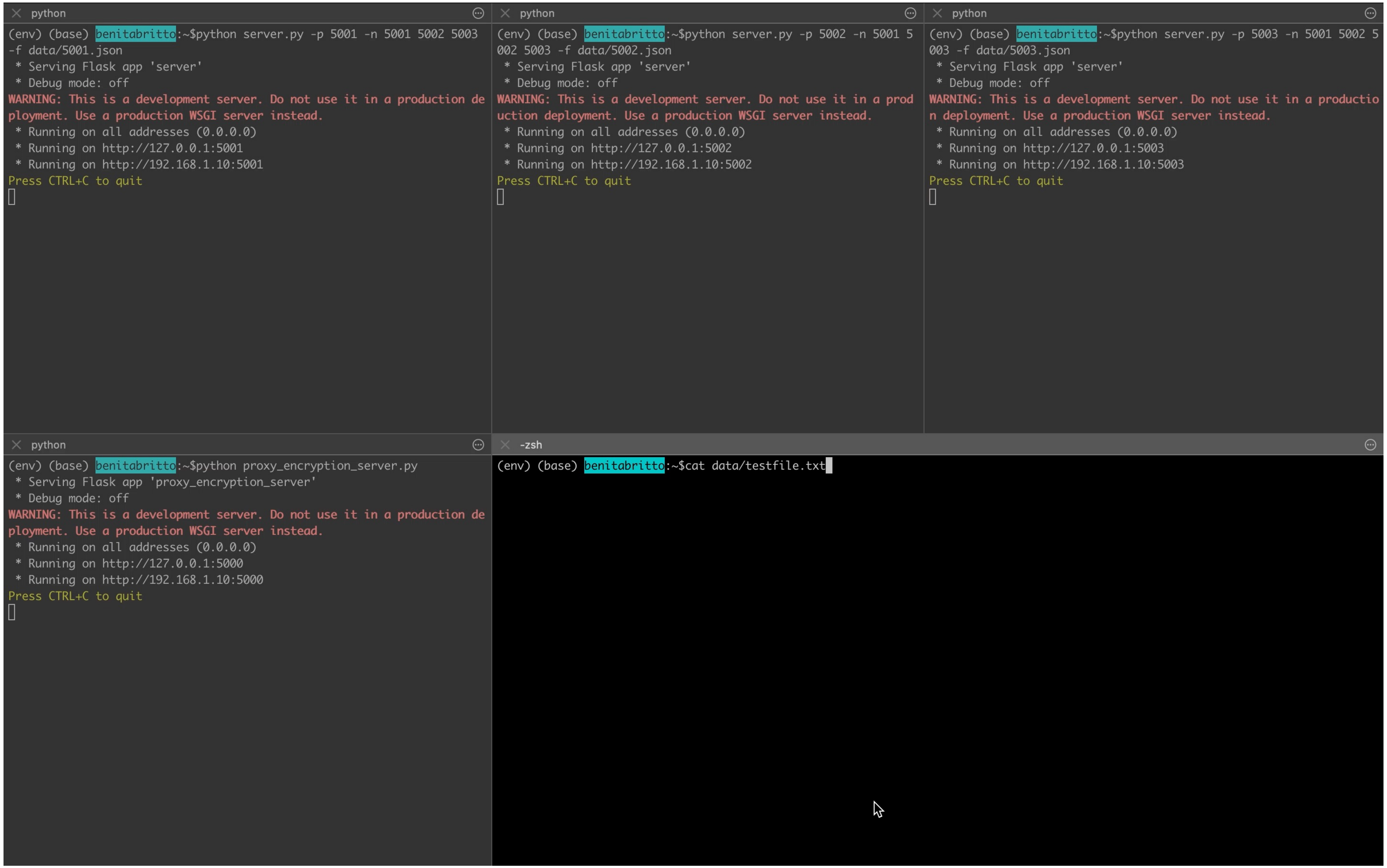


ANONYMITY



Sender: Ring Signature
Receiver: -
Data:
    capsule
    ciphertext

Sender: Ring Signature
Receiver: Stealth_Address_Ktxn(Bob)
Data:
    E_Ktxn(Txn Reference)
    E_Ktxn(Alice's Recrypt Public Key)
    E_Ktxn(Alice's Recrypt Verify Key)
    K_txn_partial = rG



python

```
(env) (base) benitabritto:~$python server.py -p 5001 -n 5001 5002 5003
-f data/5001.json
 * Serving Flask app 'server'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production de
ployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5001
 * Running on http://192.168.1.10:5001
Press CTRL+C to quit
```

python

```
(env) (base) benitabritto:~$python server.py -p 5002 -n 5001 5
002 5003 -f data/5002.json
 * Serving Flask app 'server'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a prod
uction deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5002
 * Running on http://192.168.1.10:5002
Press CTRL+C to quit
```

python

```
(env) (base) benitabritto:~$python server.py -p 5003 -n 5001 5002 5
003 -f data/5003.json
 * Serving Flask app 'server'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a productio
n deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5003
 * Running on http://192.168.1.10:5003
Press CTRL+C to quit
```

python

```
(env) (base) benitabritto:~$python proxy_encryption_server.py
 * Serving Flask app 'proxy_encryption_server'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production de
ployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.10:5000
Press CTRL+C to quit
```

-zsh

```
(env) (base) benitabritto:~$cat data/testfile.txt
```

CONCLUSION

- We implemented a **functional document sharing blockchain prototype** that:
  - securely encrypts document data and allows sharing to many users with a single copy of the document
  - preserves sender and receiver anonymity

- What we learned:
  - adding privacy to a blockchain is very difficult (when the only secret is your secret key)
  - scaling the project up would be very hard
    - we still rely on semi-trusted proxies and difficult cryptography

THANK YOU!