CryptoLife Hackathon

Prague, 26–28 October, 2018

Problem

Party A (*The Client*) wants to buy goods/services from party B (*The Supplier*). At least one of the parties cares about the privacy and wants to use payment method similar to cash. In the same time the parties care about security and want to understand the “reputation” (deal history record) of the party to interact with.

Solution

A peer-to-peer marketplace on top of status.im platform, providing a chosable tradeoff between privacy and security.

Happy path:

Client Workflow

1. The Client opens the iScream extension for Status messenger.
2. The Client sees a map.
3. The Client drops a pin on the map where they want to announce a contract for services (the Scream).
4. The Client creates a Scream with the following details:
    a. Selected cryp account for paying to the Supplier;
    b. Order details;
    c. Price offer;
    d. Timeout for the Scream;
    e. Timeout for order delivery.
5. The Client waits for a Supplier to pick up the Scream.
6. The Client receives a notification when a Supplier picks up the Scream
7. The Client and Supplier negotiate Scream details not mentioned publicly.
8. The Client waits for Scream fulfillment.
9. The Client confirms Scream fulfillment.
10. Tokens from Client are transferred to Supplier’s account.
11. Client’s reputation is increased with 1 payment point and 1 honesty point

Supplier Workflow

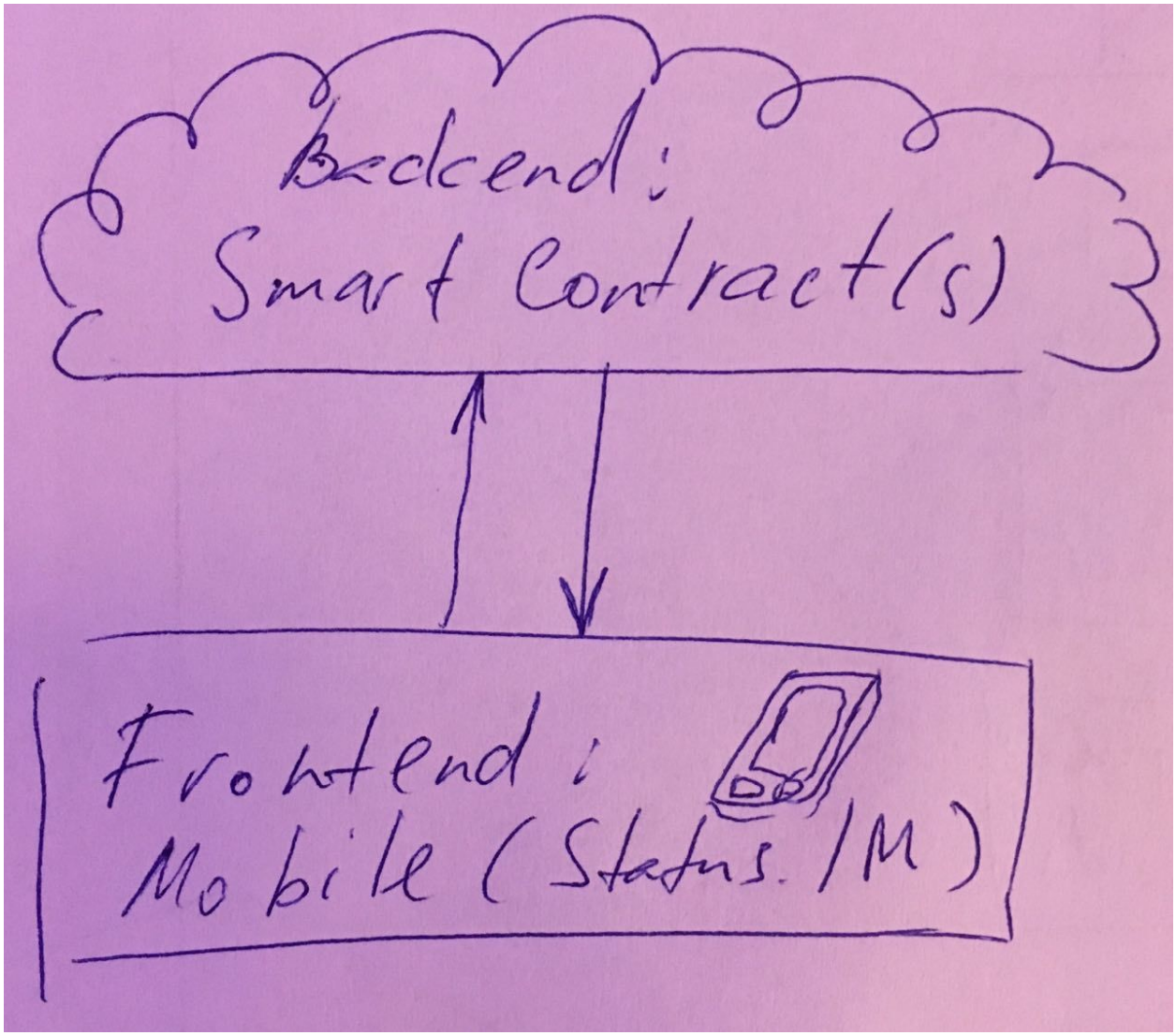
1. The Supplier opens the iScream extension for Status messenger.
2. The Supplier sees a map with Screams.
3. The Supplier selects a Scream.
4. Scream details are shown to the Supplier.
5. The Supplier responses to the Scream.
6. The Client and Supplier negotiate Scream details not mentioned publicly.
7. The Supplier fulfills the conditions of the Scream.
8. The Supplier waits for Scream fulfillment confirmation.
9. Tokens from Client are transferred to Supplier’s account.
10. Supplier’s reputation is increased with 1 payment point and 1 honesty point

Application Architecture

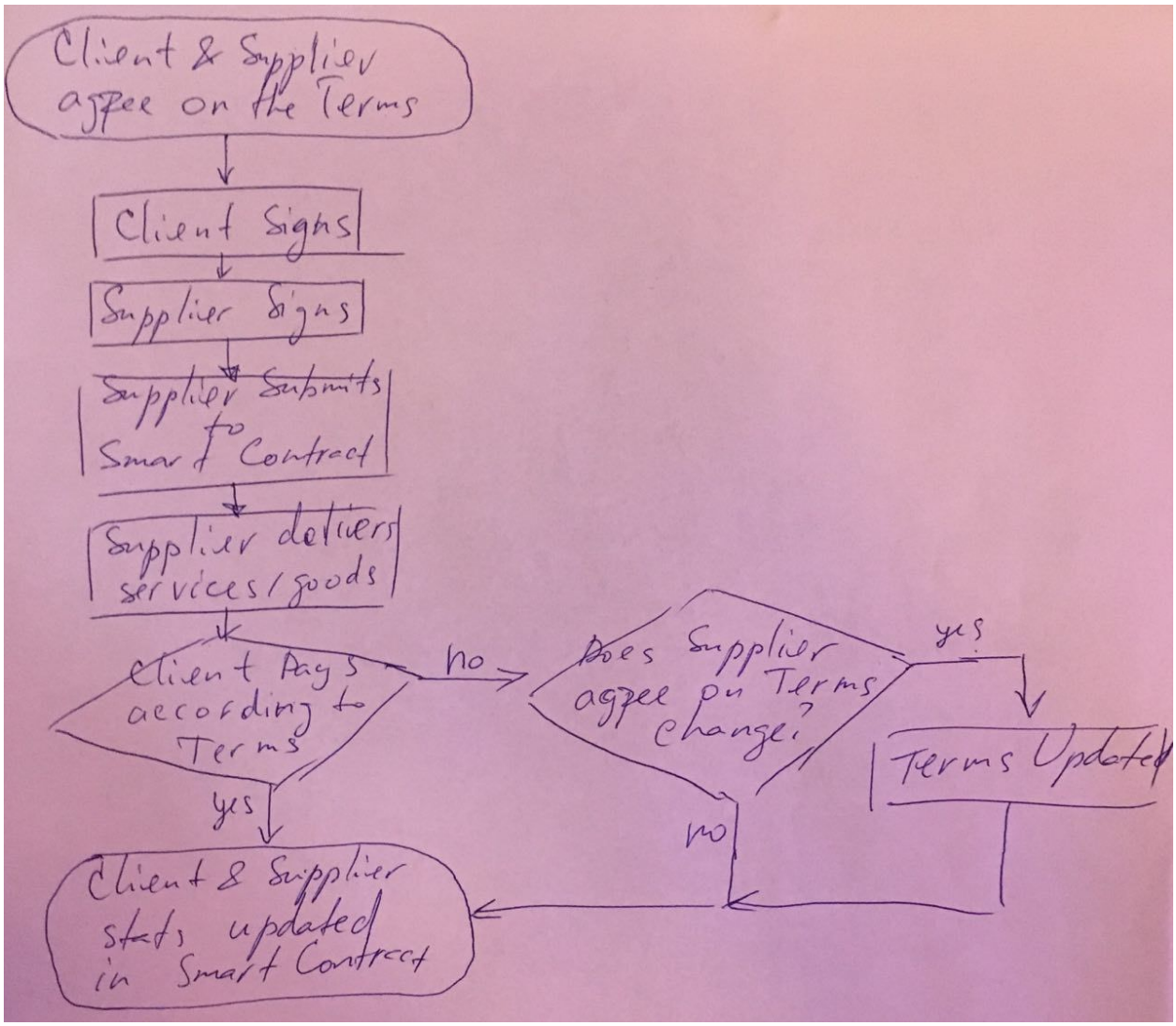
Data Model

```
/// @dev Client wants to buy goods/services and pay for them
/// @dev Cannot be zero
address client;

/// @dev Supplier wants to sell goods/services and get paid for them
/// @dev Cannot be zero
address supplier;

/// @dev Amount to pay to supplier by client, agreed initially
/// @dev Cannot be zero
uint64 initialAmount;

/// @dev Amount actually paid by client.
/// @dev Initially zero
uint64 amountPaid;

/// @dev If amount paid is less than initial amount,
///      supplier may reduce the amount to pay or leave it equal to initial
/// @dev Initially zero
uint64 amountAgreed;

/// @dev Deadline for the contract to be signed by parties
/// @dev Cannot be zero
uint64 signDueDate;

/// @dev Date the contract is signed, unix timestamp
/// @dev Cannot be zero, must be less or equal to `dueDate`
uint64 dateSigned;

/// @dev Date the contract is paid by the client, unix timestamp
/// @dev Initially zero, can remain zero
uint64 datePaid;

/// @dev Date the parties finished their interaction through the contract
/// @dev Initially zero
uint64 dateFinished;

/// @dev Time for the client to pay according to the contract
///      after the contract has been signed
/// @dev Cannot be zero
uint32 validityPeriod;

/// @dev Time for the supplier to agree/disagree on the amount
///      paid by client if it differs from initial amount agreed
/// @dev Cannot be zero
uint32 updatePeriod;
```

Module Structure



Workflow


Client & Supplier agree on the Terms
Client Signs
Supplier Signs
Supplier Submits to Smart Contract
Supplier delivers services/goods
Client Pays according to Terms
no
Does Supplier agree on Terms change?
yes
Terms Updated
yes
no
Client & Supplier stats updated in Smart Contract