# PAYMENT PROJECT

Principal Investigator: Prof. Steven Murdoch

Research Fellow: Aydin Abadi

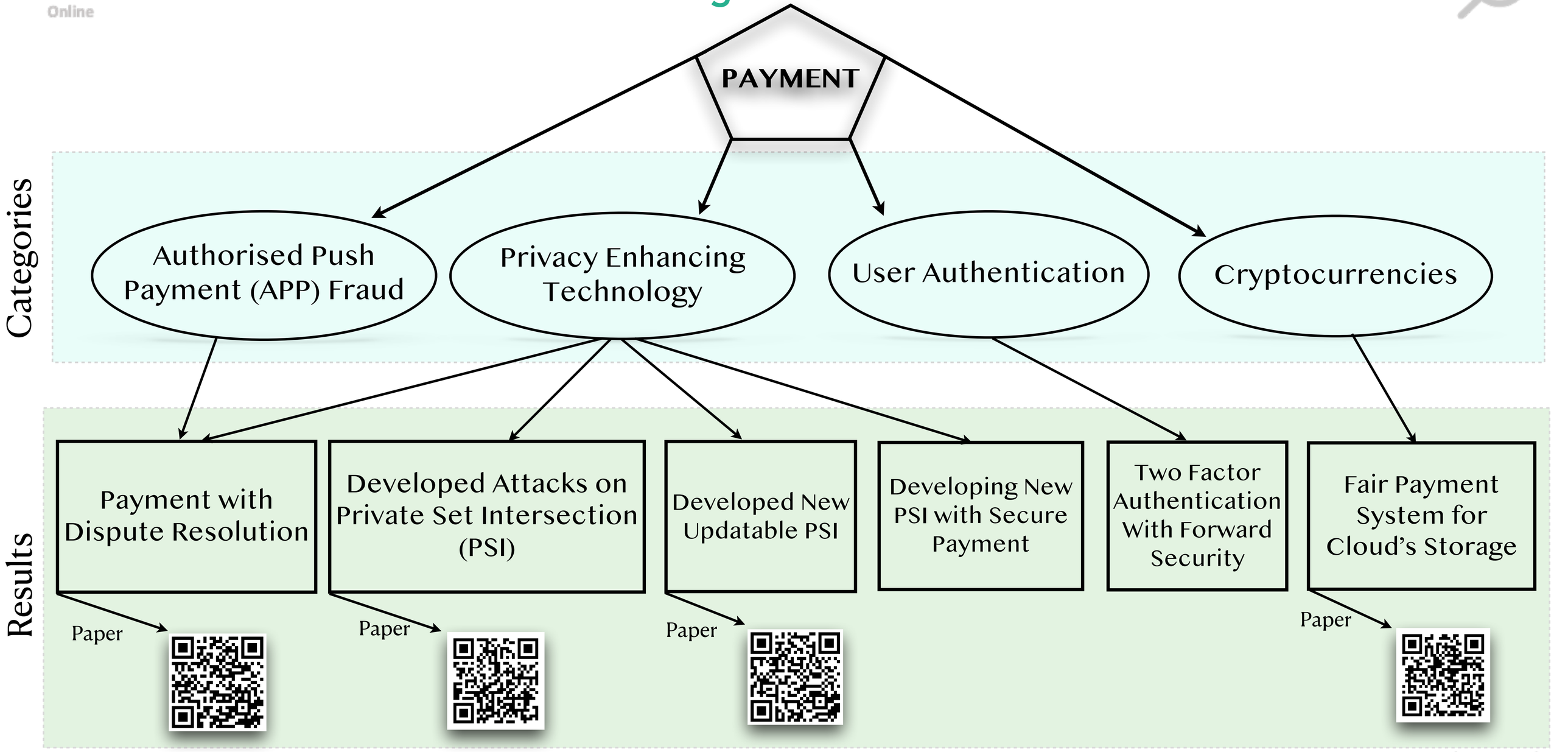
UCL

National Research Centre on Privacy, Harm Reduction and Adversarial Influence Online

# PAYMENT PROJECT

## Progress So Far

REPHRAIN
Protecting citizens online



# PAYMENT PROJECT

## Authorised Push Payment (APP) Fraud

Background



- ‘’Authorised Push Payment” (APP) fraud:
  - Definition: An APP fraud is a type of cyber-crime where a fraudster tricks a victim into making an authorised online payment into an account controlled by the fraudster.
  - It is called “authorised” because the victim authorises the payment.
  - The APP fraud has various variants, such as:
    - romance
    - investment
    - CEO
    - invoice

# PAYMENT PROJECT

## Authorised Push Payment (APP) Fraud

Background



- The amount of money lost due to APP frauds is substantial
    - Only in the first half of 2021, a total of £355 million was lost to APP frauds.
- APP fraud is a global issue.
    - According to the **FBI**'s report, victims of APP frauds reported at least a total of $419 million losses, in 2020.
    - Recently, **Interpol** warned its member countries about a concerning variant of APP fraud called investment fraud via dating software.

# PAYMENT PROJECT

## Authorised Push Payment (APP) Fraud



### Problem

- Although the UK's regulators (unlike other countries) have provided specific guidelines to financial institutes to prevent APP frauds occurrence and improve victims' protection, these guidelines are:
    - ambiguous
    - open to interpretation
- There exists no mechanism in place via which honest victims can prove their innocence.
- To date, the APP fraud **problem has been overlooked** by the information security and cryptography research communities.

# PAYMENT PROJECT

## Authorised Push Payment (APP) Fraud

Our Solution-key contributions



- To facilitate the compensation of APP frauds victims, we :

1. proposed a **new protocol** called “Payment with Dispute Resolution” (PwDR).

2. **formally defined** PwDR.

   - Identified its core security properties:

     (i) security against a malicious victim.

     (ii) security against a malicious bank.

     (iii) privacy.

3. **formally proved** the security of PwDR.

National Research Centre on Privacy, Harm Reduction and Adversarial Influence Online

# PAYMENT PROJECT

## Authorised Push Payment (APP) Fraud

Our Solution's Features

REPHRAIN Protecting citizens online

- The PwDR offers **transparency** by
  - (1) accurately formalising reimbursements' conditions
  - (2) offering traceability
  - (3) providing an evidence-based final decision
- The PwDR offers **accountability**, as it is equipped with auditing mechanisms that help identify the party liable for an APP fraud loss.
  - The auditing mechanisms themselves are accompanied by our lightweight **privacy-preserving threshold voting** protocols.
    - Our voting protocols let auditors vote privately without having to worry about being retaliated against, for their votes.

.

# PAYMENT PROJECT

## Authorised Push Payment (APP) Fraud

Our Solution's Features



The PwDR is efficient:

- We analysed the PwDR's cost via both:
  - asymptotic analysis
  - concrete evaluation
- our analysis indicates the protocol is highly efficient.

# PAYMENT PROJECT

## Protecting Victims of APP Frauds

### The PwDR Protocol's Cost



Asymptotic cost analysis

| Party | Setting | | Computation Cost | Communication Cost |
|---|---|---|---|---|
| | $e = 1$ | $e > 1$ | | |
| Customer | ✓ | ✓ | $O(1)$ | $O(1)$ |
| Bank | ✓ | ✓ | $O(1)$ | $O(1)$ |
| Arbiter $\mathcal{D}_1, \ldots, \mathcal{D}_{n-1}$ | ✓ | ✓ | $O(1)$ | $O(1)$ |
| Arbiter $\mathcal{D}_n$ | ✓ | | $O(n)$ | $O(1)$ |
| | | ✓ | $O(\sum_{i=e}^{n} \frac{n!}{i!(n-i)!})$ | $O(\sum_{i=e}^{n} \frac{n!}{i!(n-i)!})$ |
| Dispute resolver | ✓ | ✓ | $O(n)$ | $O(1)$ |

Concrete cost analysis

| Party | $n = 6$ | | $n = 8$ | | $n = 10$ | | $n = 12$ | |
|---|---|---|---|---|---|---|---|---|
| | $e = 1$ | $e = 4$ | $e = 1$ | $e = 5$ | $e = 1$ | $e = 6$ | $e = 1$ | $e = 7$ |
| Arbiter $\mathcal{D}_n$ | 0.019 | 0.220 | 0.033 | 0.661 | 0.035 | 2.87 | 0.052 | 10.15 |
| Dispute resolver $\mathcal{DR}$ | 0.001 | 0.015 | 0.001 | 0.016 | 0.001 | 0.069 | 0.003 | 0.09 |

*

Time in millisecond

n: number of arbiters

e: threshold

# PAYMENT PROJECT

## Protecting Victims of APP Frauds

Main Tools We Used



- The PwDR Protocol's building blocks:
  - Commitment scheme
  - Digital signature
  - Smart contract and blockchain
  - Pseudorandom function
  - Bloom filter
  - Threshold voting protocols

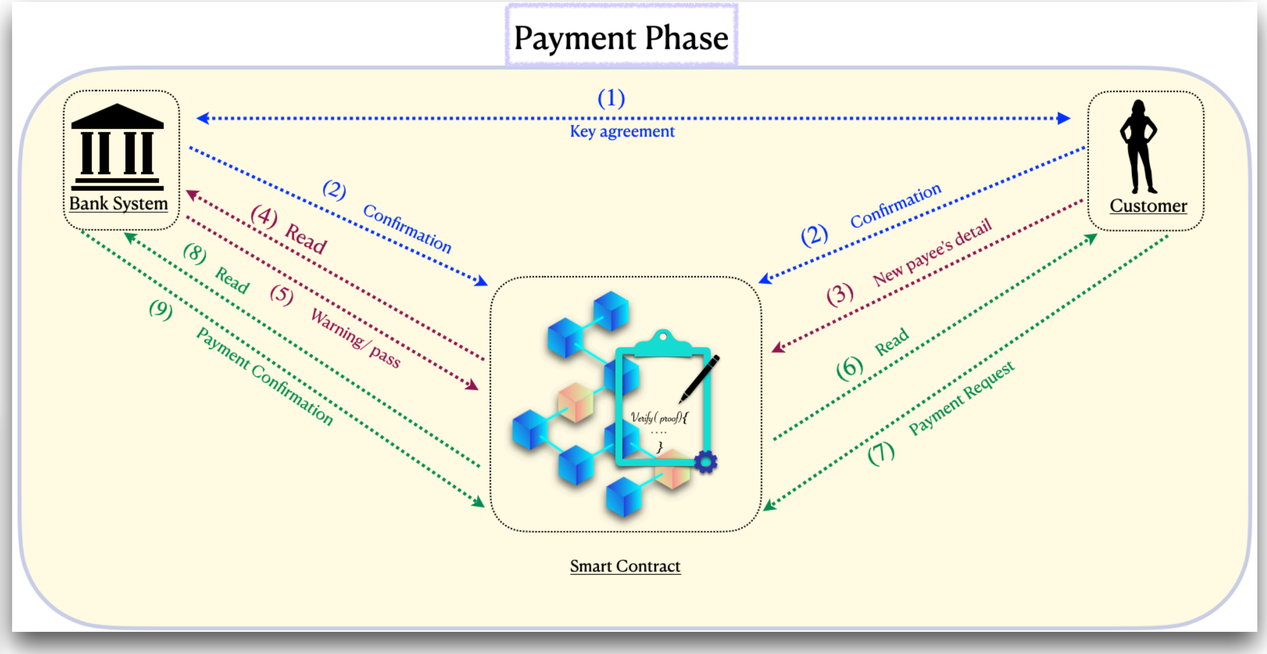
# PAYMENT PROJECT

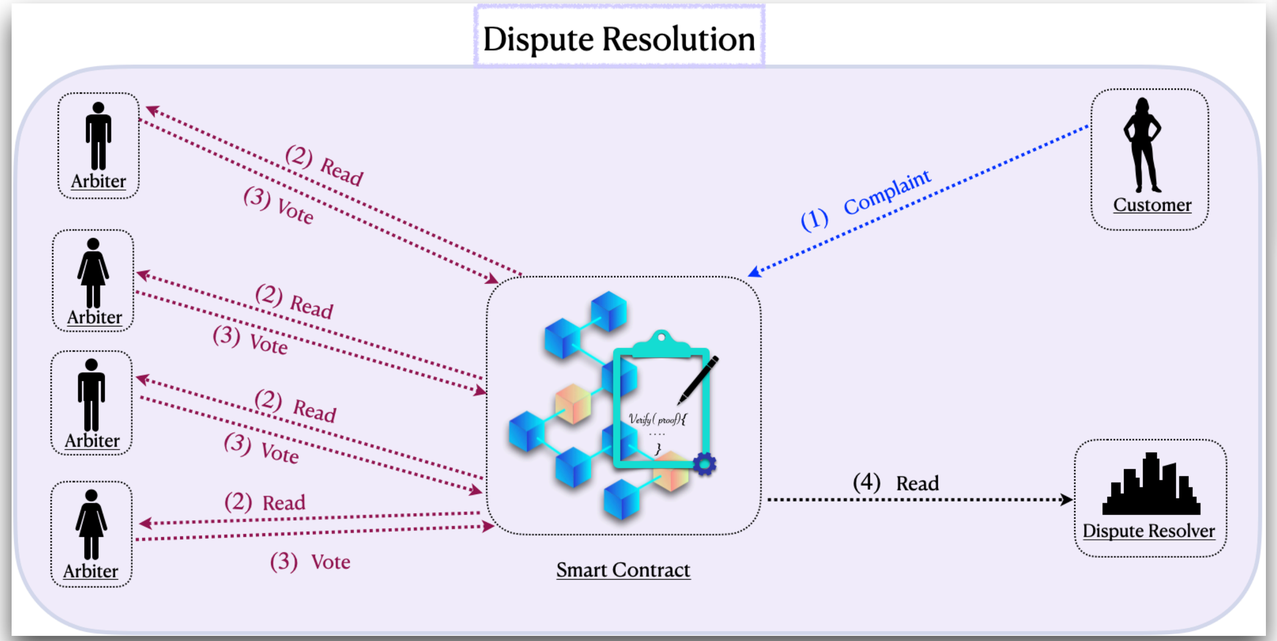
## Protecting Victims of APP Frauds

### The PwDR Protocol's Workflow



- The PwDR Protocol involves two main phases:
  - Payment
  - Dispute resolution



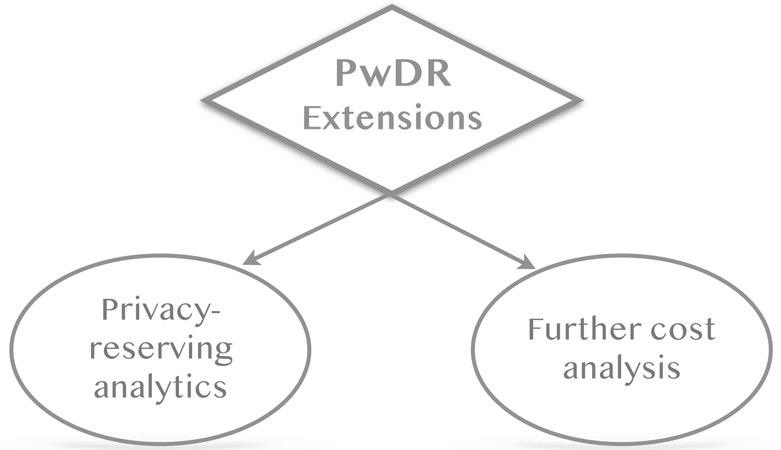



# PAYMENT PROJECT

## Protecting Victims of APP Frauds

Extension and Further analysis of PwDR






- We have been:

1. extending the PwDR's functionality
    - developing new privacy-preserving analytics.
2. further analysing the PwDR's performance (in collaboration with Dr. Partha Das Chowdhury form the University of Bristol):
    - Implemented the PwDR's smart contracts.
    - Analysed its costs.

# PAYMENT PROJECT

## Protecting Victims of APP Frauds

Implementation of smart contracts



Now Dr. Partha Das Chowdhury

will discuss the implementation of the smart contracts

National Research Centre on
Privacy, Harm Reduction
and Adversarial Influence
Online

# PAYMENT PROJECT

## Protecting Victims of APP Frauds

REPHRAIN
Protecting citizens online

**Architecture**

Smart Contracts
1. SAP – Key Management
2. Add Payee
3. Generate Payment Request
4. Make Payment
5. Generate Compliant Request
6. Verify Key Agreement
7. Resolve Complaint

Org1.peer Org2.peer Org3.peer Org4.peer Org5.peer Org6.peer Org7.peer

Docker Containers

Hyperledger Fabric – 2.2.3

Ubuntu 20.04.3 LTS - AWS

SevenOrgsChannel:
<configtx.yaml>

Consortium:
SampleConsortium

- <<: *ChannelDefaults
- Application:
- <<: *ApplicationDefaults
- Organizations:
- - *Org1 - Bank
- - *Org2 - Account Holder
- - *Org3 - FCA
- - *Org4 - Which
- - *Org5 - Arbitrator
- - *Org6 - Arbitrator
- - *Org7 -Arbitrator
- Capabilities:
- <<: *ApplicationCapabilities

# PAYMENT PROJECT

## Protecting Victims of APP Frauds



### Commands

./network.sh up

./network.sh createChannel -c drchannel -verbose

./network.sh deployCC -c drchannel -ccn sap -ccl go -ccv 0.1 -ccs 1 -ccp /home/ubuntu/dispute-resolution/sap -ccep "AND(\"Org2MSP.peer\")"
./network.sh deployCC -c drchannel -ccn payee -ccl go -ccv 0.2 -ccs 1 -ccp /home/ubuntu/dispute-resolution/payee -ccep "AND(\"Org1MSP.peer\")"
./network.sh deployCC -c drchannel -ccn payment -ccl go -ccv 0.1 -ccs 1 -ccp /home/ubuntu/dispute-resolution/payment -ccep "AND(\"Org1MSP.peer\")”
./network.sh deployCC -c drchannel -ccn complaint -ccl go -ccv 0.1 -ccs 1 -ccp /home/ubuntu/dispute-resolution/complaint -ccep "OR(\"Org1MSP.peer\",\"Org5MSP.peer\",\"Org6MSP.peer\",\"Org7MSP.peer\")"

./scripts/invoke-fcn.sh drchannel
./scripts/query-fcn.sh drchannel complaint

# PAYMENT PROJECT

## Protecting Victims of APP Frauds



### Lines of Code (LoC)

| Sl No | Particular | LoC – Without Privacy | LoC - Privacy |
|---|---|---|---|
| 1 | SAP - Chaincode | - | 346 |
| 2 | Payee - Chaincode | 291 | 376 |
| 3 | Payment - Chaincode | 393 | 478 |
| 4 | Complaint - Chaincode | 690 | 775 |
| 5 | Helper Functions – common for every Chaincode | 85 | 85 |
| 6 | Encryption – common for every Chaincode, except SAP | - | 117 |
| | Total LoC | 1559 | 2177 |

# PAYMENT PROJECT

## Protecting Victims of APP Frauds



### Test Conditions

```
async submitTransaction() {


    const complaint = queue.nextComplaint()
    const resolveType = helper.getRandomResolveType()
    const K1 = helper.sapKeys.K1
    const K2 = helper.sapKeys.K2

    let args = {
        contractId: 'complaint',
        contractVersion: '1.1',
        contractFunction: 'ResolveComplaint',
        contractArguments: [complaint.ID, resolveType, K1, K2],
        timeout: 60,
    };
```

Smart Contract

```
- label: Resolve Complaint
  txDuration: 120
  rateControl:
      type: fixed-load
      opts:
        transactionLoad: 100
  workload:
    module: benchmarks/dispute-resolution-encrypted/ResolveComplaint.js
```

Test Iterations

# PAYMENT PROJECT

## Protecting Victims of APP Frauds



**Test Report**

HYPERLEDGER CALIPER

### Basic information

DLT: fabric

Name:

Description:

Benchmark Rounds: 5

Details

### Benchmark results

Summary

Generate Payment Request

Make Payment

Generate Complaint

Resolve Complaint

Verify Agreement

# Caliper report

## Summary of performance metrics

| Name | Succ | Fail | Send Rate (TPS) | Max Latency (s) | Min Latency (s) | Avg Latency (s) | Throughput (TPS) |
|---|---|---|---|---|---|---|---|
| Generate Payment Request | 1918 | 0 | 15.6 | 9.33 | 0.43 | 4.56 | 15.5 |
| Make Payment | 1918 | 0 | 16.5 | 10.09 | 0.44 | 3.90 | 16.1 |
| Generate Complaint | 2016 | 0 | 16.6 | 8.26 | 0.41 | 4.02 | 16.3 |
| Resolve Complaint | 2117 | 107 | 17.7 | 7.57 | 0.26 | 3.73 | 17.7 |
| Verify Agreement | 18700 | 0 | 156.8 | 1.02 | 0.01 | 0.26 | 156.8 |

# The end