# Secure IoT applications using scalable Blockchain Models and PQ Primitives

Muhammad Rameez

# Table of Contents

## 2. Introduction

Blockchain has exploded as the technology of the future for several use cases including Cross border payments, Peer to Peer transactions, Regulatory Compliance, Healthcare and Supply Chain Management. It provides a tamper proof immutable ledger which can be particularly useful in tracking goods and services as they move and change hands in the supply chain. It enables new and innovative means of organizing and tracking data. Smart Contracts are one of the killer applications of the Blockchain. They have the potential to revolutionize and automate several industries including supply chain management. Smart Contract platforms such as the Ethereum can use tracking data to automate various functions and events in the supply chain life cycle. Ethereums distributed ledger also provides total transparency to all parties involved. In this Thesis I explore an Enterprise level solution for supply chain management and tracking using the Ethereum Blockchain. I will also use Raiden based state channels for efficiently establishing payment channels between an organization and its suppliers.

## 3. Description

In order to better explain the use case, let us consider an organization that has several suppliers around the world. It uses these suppliers to deliver different components to its factory floor. The components must be delivered on time and have to be stored, handled and transported under specific physical or environmental constraints. The Organization and Suppliers set a delivery date and agree upon the specifics of shipping and handling constraints in advance. This agreement is deployed in the form of a smart contract on the Ethereum blockchain, once it is deployed its terms cannot be altered. The terms of the contract can include delivery deadline, handling constraints like exposure to high temperature, pressure, air etc. The Smart contract can detect any breaches of the agreed upon terms and take appropriate actions to compensate or penalize the parties involved. Payment channels are established between the organization and its suppliers to handle payments efficiently. The payment channels are deployed using Raiden, an off chain state channel solution developed by Brainbot technologies. For detailed description about Raiden and state channels please see sections [], [], []. The state of the art for our use case are the smart contracts for monitoring supply chain cycle and the decentralized application developed to interact with smart contracts and the payment channels. This sample use case is modeled in figure 1.

## 4. System Architecture

The system for our sample use case has four main components 1. Smart contracts 2. Payment Channels 3. Decentralized Apps and 4. Smart packages. This system is modeled in Figure 1; In this case the organization has two suppliers supplying two different types of parts. Each component has different requirements for how it should be handled during shipping. These requirements are instantiated in the two smart contracts i.e. Smart contract A and Smart Contract B which will be referred to as SC-A and SC-B from here on. The company and its suppliers agree upon payment structure along with the terms and conditions for shipping and handling in advance. Smart contracts insure trustless compliance of the agreed upon terms and conditions. Payment channels are established between the organization and its suppliers to insure frictionless payments and remunerations. In this case two payment channels exist Payment Channel A between organization and Supplier A and Payment Channel B between organization and supplier B. These will be referred to as PC-A and PC-B from here on. The system can further be extended to have payment channels between suppliers and shippers shown in the figure as Payment Channels C and D and which will be referred to as PC-C and

PC-D. The organization is responsible for compensating every actor in the supply chain life cycle. In order to do so it can either use Ethereum or issue its own ERC20 tokens which can be exchanged for Eth using a smart contract. We will consider the case that organization is issuing ERC20 tokens which will be used for transferring value between actors in the supply chain cycle. The main advantage of issuing your own ERC20 token is that you can fix the exchange rate so it's always tied to a fixed Fiat value. This will protect against price fluctuations in cryptocurrency value.  All payment channels for a particular supply chain cycle use the same tokens. The organization establishes a fixed exchange rate for its ERC20 tokens i.e. each token is worth 1$ and blocks enough Eth in an external smart contract based escrow to cover all expenses incurred. The suppliers and shippers get payed in ERC20 tokens which they can send to the smart contact to get Eth.

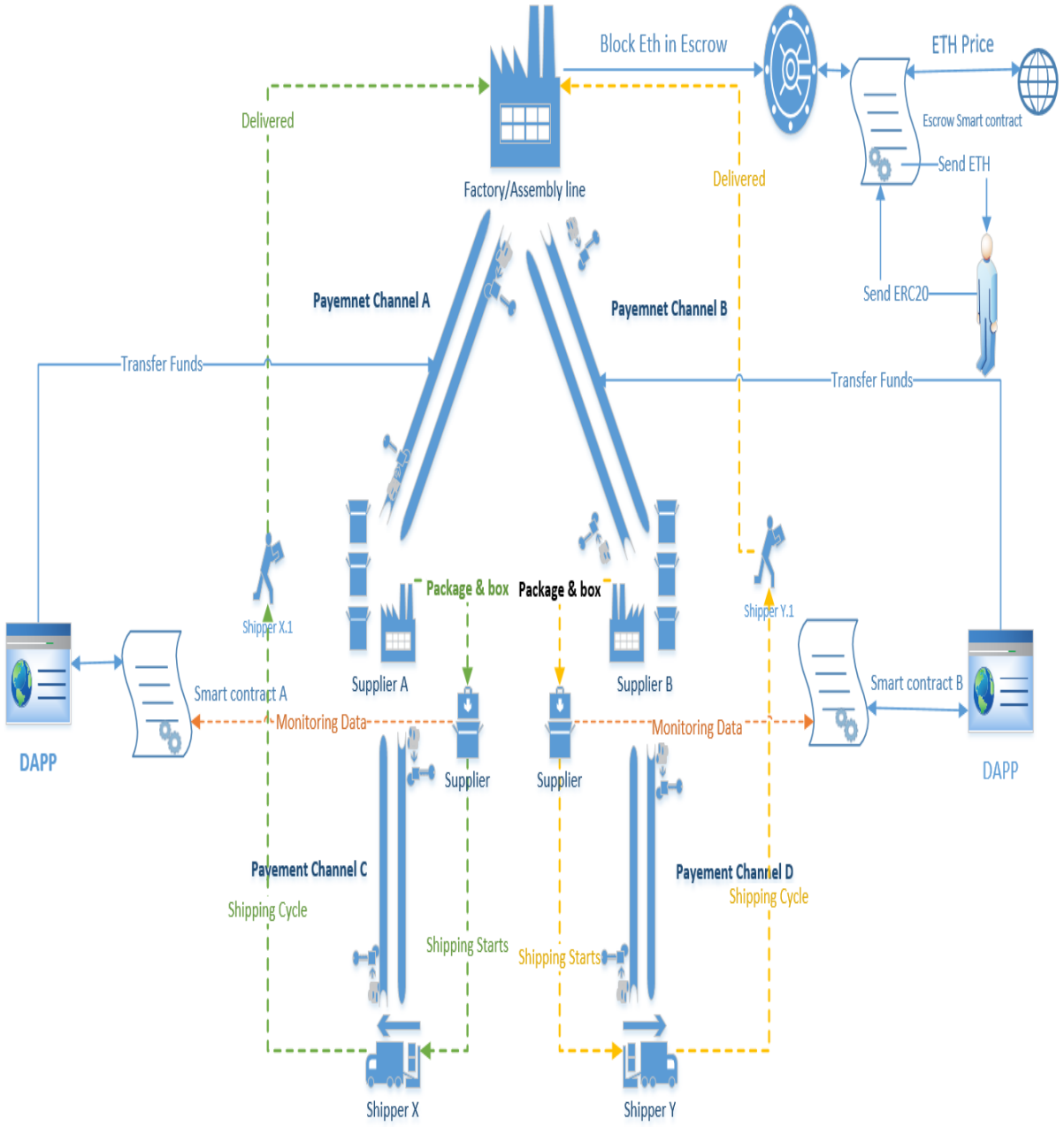


Figure 1

## 4.1. Supply chain Life Cycle

In order to better explain the supply chain life cycle of our particular use case we use the simplified version of the system presented in fig 1. The simplified version consists of only one supplier and shipper. The company places a new order for components with its supplier and stipulates shipping and handling constraints in the smart contract. Once the order has been placed it funds the payment channel associated with its supplier. Supplier packages the components along with a tamper proof smart device which will communicate shipping data with the smart contract. The smart device is a Raspberry Pi running a custom program to send encrypted and signed data to the smart contract. This program will be referred to as RApp from hence forth. The Rapp is also one of the state of the arts for this thesis, it can be configured to send data continuously or when special events are triggered i.e. some shipping violation has occurred. The Rapp sings the data with the key of the current shipper/ handler. We need internet connectivity throughout the shipping life cycle in order to insure monitoring data is continuously communicated with the Smart Contract on the block chain. The monitoring starts as soon as the components are packaged in the supplier warehouse, at this time the data sent to the SC-A is signed with key of the supplier. When the shipper X takes possession of the package from the supplier they scan the package. This scanning event represents changing of ownership of responsibility in the supply chain cycle which means from this point on all data sent to the smart contract will be signed with the key of the shipper X. If there is more than one shipper as show in Fig 2 each one scans the package to take over shipping responsibilities. In our system Smart Contracts and Dapps are responsible for catching violations and taking appropriate actions to penalize violators and compensate aggrieved parties. This brings full transparency for all stake holders in the supply chain life cycle and resolved disputes if any in an efficient and trustless manner. If the package is delivered to the factory without incurring any violations the company releases the funds in the payment channel and everyone gets paid. If any violations occur the DApp/SC determines the party responsible for the violations and takes appropriate actions. These actions could range from giving the supplier or shipper a bad rating to monetary punishments such as withholding payments etc.

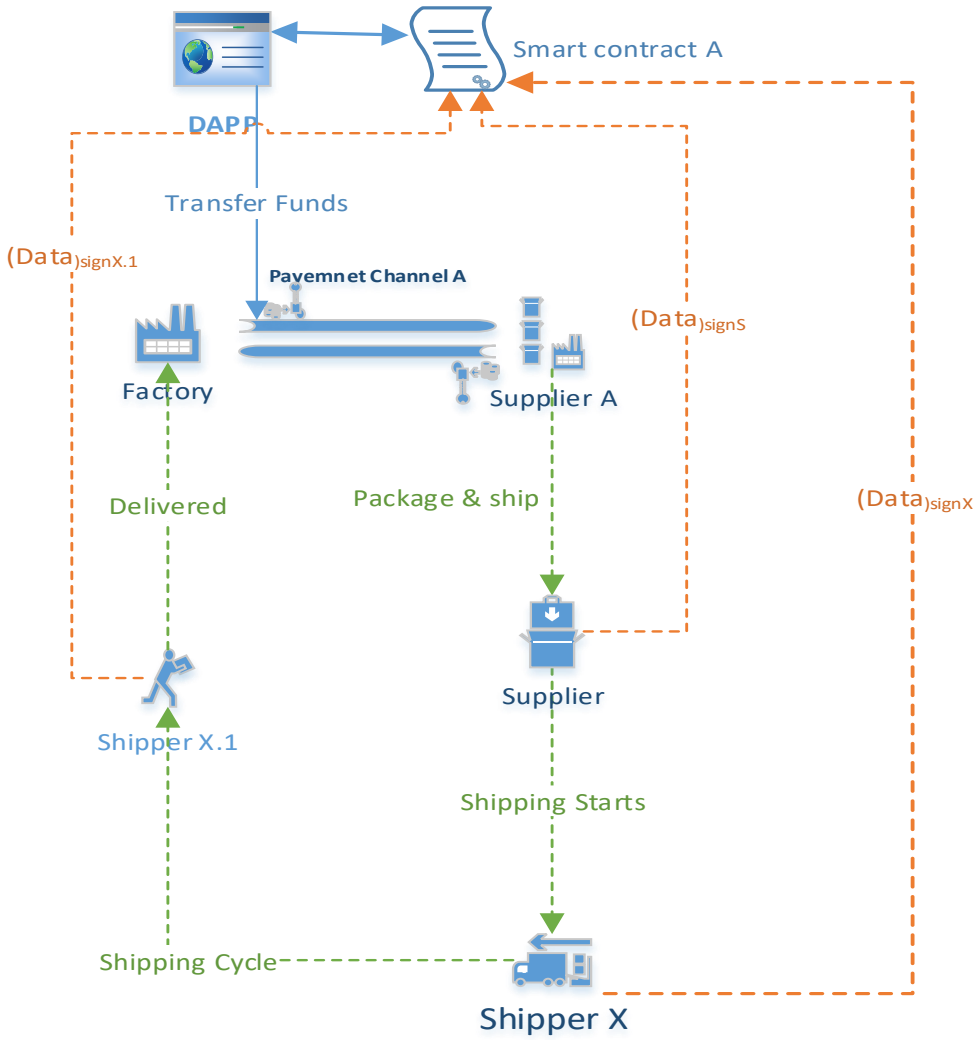


Figure 2

### 4.1.1. Signing Data Vs Signed Cookie

**Signing Data**

Initially the idea was to Send signed monitoring data to the smart contract and Dapp i.e. the data sent by the raspberry pi will be signed by each shipper's private key. The simplest way to do this would be to store every shippers private key in the raspberry pi. This design choice has several limitations. 1. private keys should only ever be in possession of one person i.e. the shipper so that there is no dispute on the identity of the person, entity of company controlling the private key. 2. This would severely restrict the number of different shippers that can be involved in the supply cycle i.e. no shipper whose private key has not been stored in the raspberry pi can ever handle the package no matter the circumstances. One possible work around could be to use multi signature design where each entity has two or more keys. One key is stored in the Raspberry Pi while the other is with the entity, the key stored in the Pi signs data on the behalf of the entity currently in possession of the package. In order to unlock funds, the entity requires both keys i.e. the key they transferred to the PI and the second key which they had with them. This solution draws inspiration from Parity techs, multi signature wallets. The feasibility of such a solution in our use case remains to be seen However, this would require

significant rework of all existing systems including Raiden payment channels and would be well beyond the scope of this thesis.

### Signed Cookies

An alternative solution can be to use signed cookies for associating identity of the shipper with monitoring data sent to the Smart Contract. Each time a shipper takes control they will need to scan the package, usually scan events are about just reading package information i.e. one-way data transfer in which scanner reads the bar code on a package. However, we can use this scan event to send some signed data to the Raspberry pi for example a signed cookie which says Bob is in possession of this packet, when the next shipper takes possession of the package they scan the package and transfer their own signed cookie to the PI which says Alice is now in possession of the package. When Raspeberry Pi sends data to the smart contract it sends data+cookie. Data is used to identify any violations while associated cookie is used to establish the identity of the shipper. Please have a look at the following diagram.

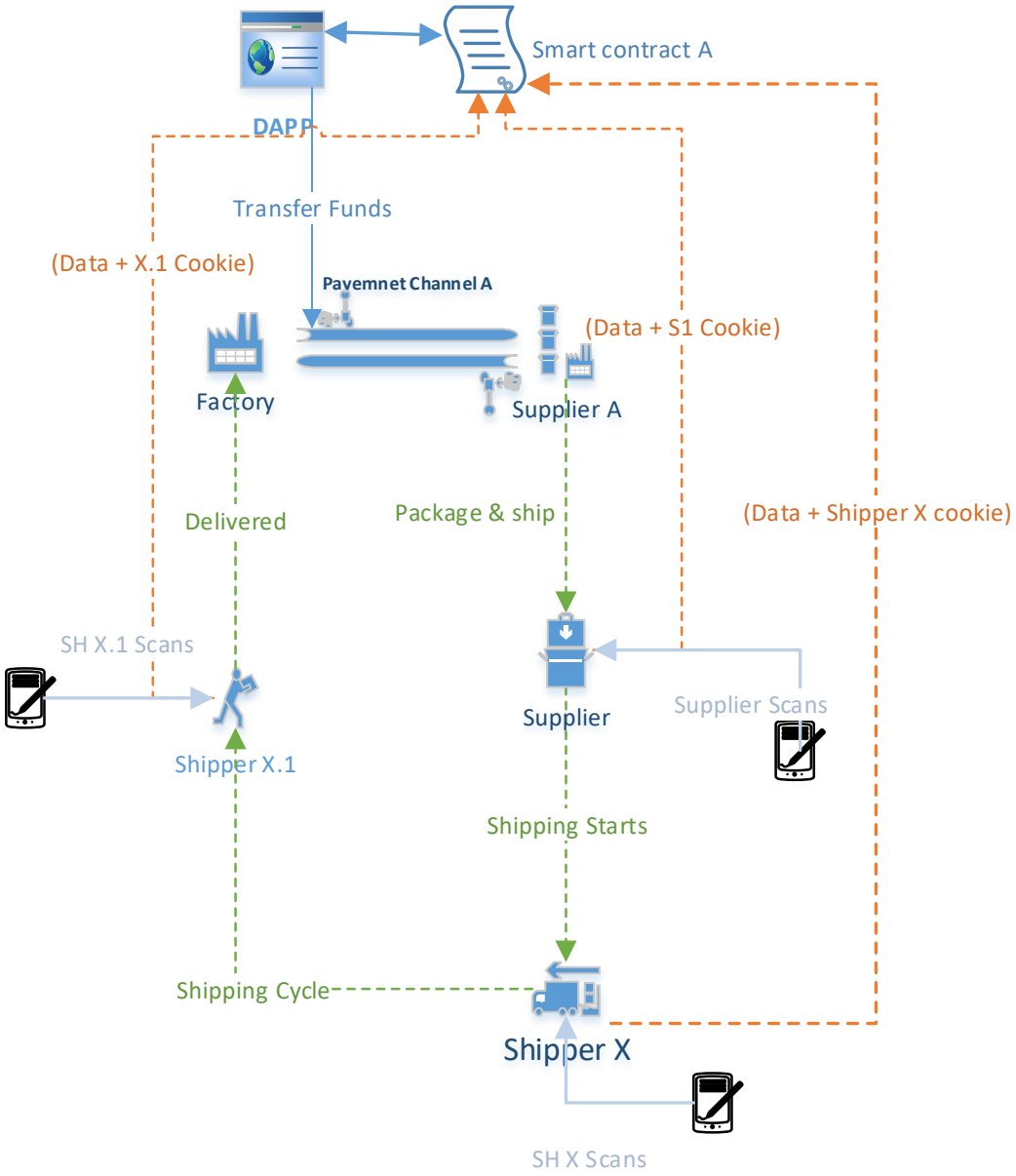


Figure 3

## 4.2. System sequence Diagram

The sequence diagram for the System which models Supply chain life cycle of our use case is shown in fig3. There are two main human actors in this system a Payer i.e. the company and the payee i.e. supplier and/or shipper. Everything starts with negotiations of terms and conditions between these two actors. The second step is to establish payment channels between the supplier and the company or to fund existing PCs. The channel is deployed using Raiden and is funded by the company. Once the terms and conditions are finalized they are published in the form of a Smart Contract on the Ethereum Blockchain. The smart contract is responsible for keeping track of shipping data and penalize any violations of the agreed upon terms and conditions. Rapp is responsible for communicating shipping and tracking data with the smart contract. It signs the data with the key of the current shipper or handler before sending it to the SC. We also have DAPP or decentralized application for reading tracking data from the smart contract and presenting it to the parties involved. The DAPP also performs different actions based on this data i.e. check for violations and display violations. If it finds any violations it takes appropriate actions to penalize the responsible shipper or supplier. The penalties can range from downgrading the rating of the partner to deducting total payable tokens. Each violation has a penalty associated with it which is stipulated in the smart contract. The Dapp checks for violations and associated penalties before releasing funds to counter parties in the payment channel. If no violations were detected all agreed upon tokens are released to the suppliers /counter party. If any violations are detected the DAPP deducts the amount associated with those violations before releasing funds. Consider the case where the package contents should never be exposed to a temperature greater than 40C. If at any point this temp limit is exceeded the responsible party must pay 10 tokens as remunerations. When the DAPP determines a temperature limit violation has occurred it automatically determines the responsible party and reduces their total payable balance by 10 tokens. When the funds are released at the end of the shipping cycle and everyone gets paid the responsible party gets 10 tokens less than the agreed upon sum.

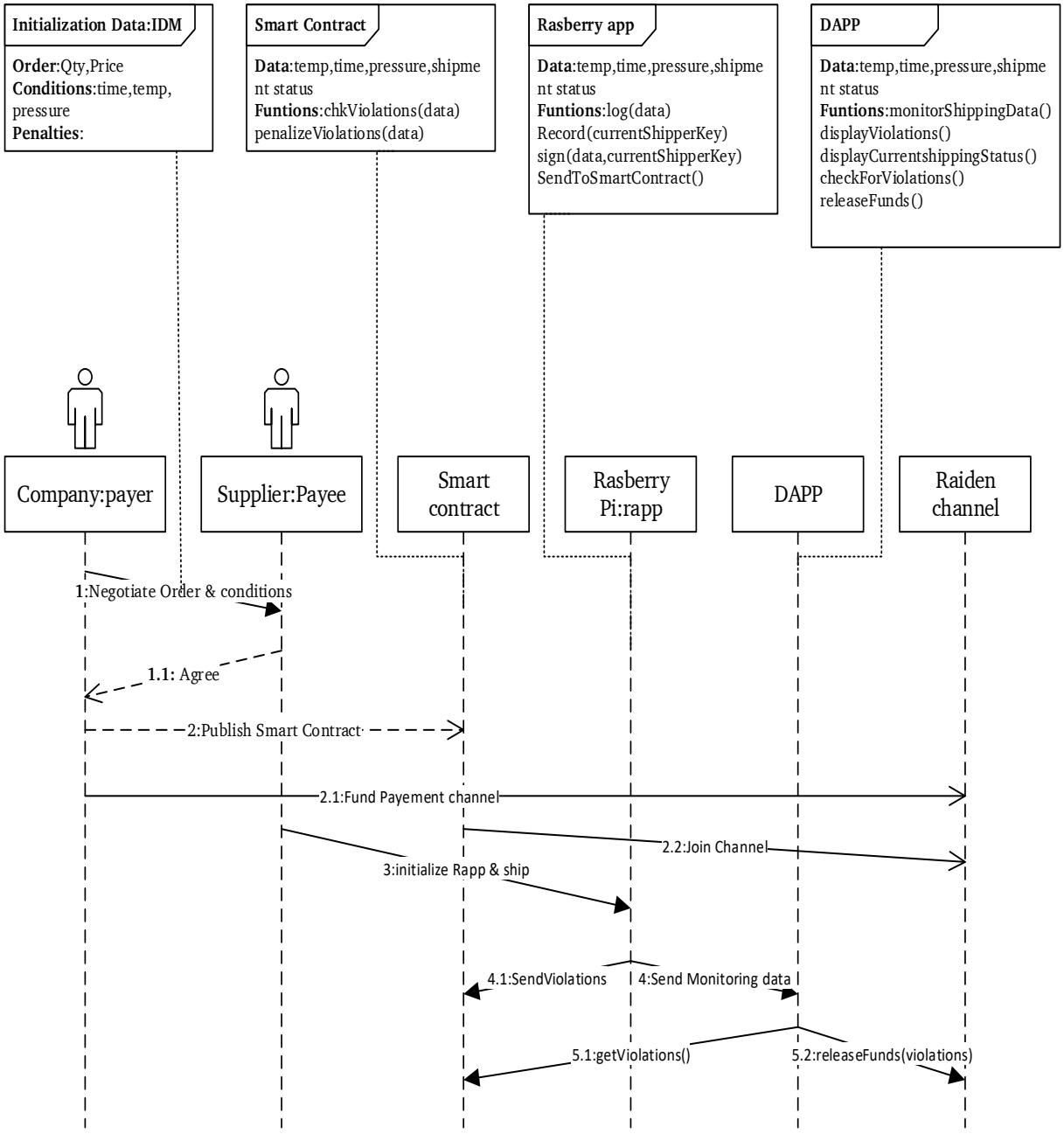


Figure 4

## 5. References

[1]