Law, Science and Technology
MSCA ITN EJD n. 814177

Mirko Zichichi[1,2], Luca Serena[2], Stefano Ferretti[3], Gabriele D'Angelo[2]

[1]Universidad Politécnica de Madrid
[2]University of Bologna
[3]University of Urbino "Carlo Bo"

Incentivized Data Mules Based on State-Channels

InDaMul

General Problem $\rightarrow$ Possible Solution

People decides to move towards *countrysides and rural areas* -> it is not possible to implement (costly) smart city services.

Smart territories:

General Problem $\rightarrow$ Possible Solution

People decides to move towards *countrysides and rural areas* -> it is not possible to implement (costly) smart city services.

Smart territories:

- no support by a *wide area network connectivity*

General Problem $\rightarrow$ Possible Solution

People decides to move towards *countrysides and rural areas* -> it is not possible to implement (costly) smart city services.

Smart territories:

- no support by a *wide area network connectivity*
- or solutions might result *too costly* (e.g. satellite connections)

General Problem $\rightarrow$ Possible Solution

People decides to move towards *countrysides and rural areas* -> it is not possible to implement (costly) smart city services.

Smart territories:

- no support by a *wide area network connectivity*
- or solutions might result *too costly* (e.g. satellite connections)
- Needed:

General Problem $\rightarrow$ Possible Solution

People decides to move towards *countrysides and rural areas* -> it is not possible to implement (costly) smart city services.

Smart territories:

- no support by a *wide area network connectivity*
- or solutions might result *too costly* (e.g. satellite connections)
- Needed:
  - *novel opportunistic solutions* -> share and reuse data, services, computation.

General Problem → Possible Solution

People decides to move towards *countrysides and rural areas* -> it is not possible to implement (costly) smart city services.
Smart territories:

- no support by a *wide area network connectivity*
- or solutions might result *too costly* (e.g. satellite connections)
- Needed:
  - *novel opportunistic solutions* -> share and reuse data, services, computation.
  - **Data Mules** (Mobile Ubiquitous LAN Extensions) ->
    even in the absence of Internet are able to collect data from sensors and to exploit their own mobility to carry the information to destination

Specific Problem

InDaMul: a dapp incentivizing participants in Data Mule-based communications, by combining Distributed Ledger Technologies (DLTs) and Decentralized File Storages (DFS):

Specific Problem

InDaMul: a dapp incentivizing participants in Data Mule-based communications, by combining Distributed Ledger Technologies (DLTs) and Decentralized File Storages (DFS):

- increasingly used to create **common, decentralized** and **trustless** infrastructure

Specific Problem

InDaMul: a dapp incentivizing participants in Data Mule-based communications, by combining Distributed Ledger Technologies (DLTs) and Decentralized File Storages (DFS):

- increasingly used to create **common, decentralized** and **trustless** infrastructure
- **smart contracts** -> ability to **automate and enforce** processes

Specific Problem

InDaMul: a dapp incentivizing participants in Data Mule-based communications, by combining Distributed Ledger Technologies (DLTs) and Decentralized File Storages (DFS):

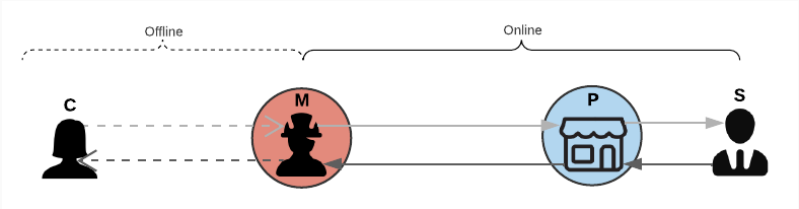
- increasingly used to create **common, decentralized** and **trustless** infrastructure
- **smart contracts** -> ability to **automate and enforce** processes
- A **state channels** are opened between participants -> ERC20 tokens deposit in a smart contract.

Specific Problem

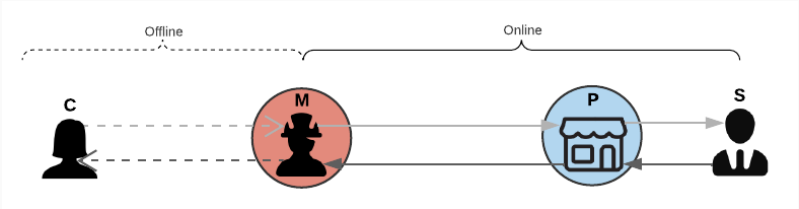
InDaMul: a dapp incentivizing participants in Data Mule-based communications, by combining Distributed Ledger Technologies (DLTs) and Decentralized File Storages (DFS):

- increasingly used to create **common, decentralized** and **trustless** infrastructure
- **smart contracts** -> ability to **automate and enforce** processes
- A **state channels** are opened between participants ->
  ERC20 tokens deposit in a smart contract.
- **State channel networks** -> participants pay by using other participants as relays.

InDaMul

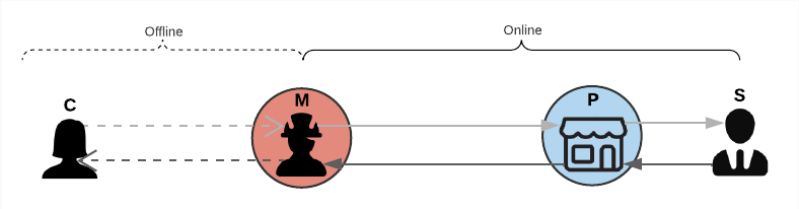


InDaMul



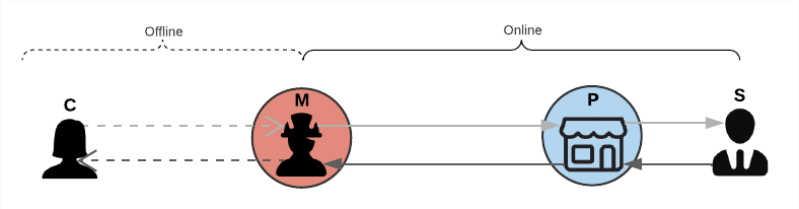
- ***Client*** (*C*) is offline and wants to send a message to a ***Server*** (*S*) that is online

InDaMul



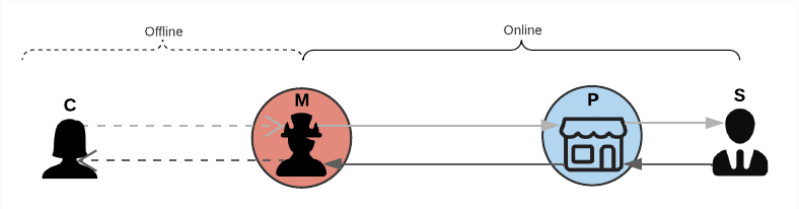
- ***Client*** (***C***) is offline and wants to send a message to a ***Server*** (***S***) that is online
- A ***Data Mule*** (***M***) retrieves (offline) **C**'s payload and brings it to a ***Proxy*** (***P***).

InDaMul



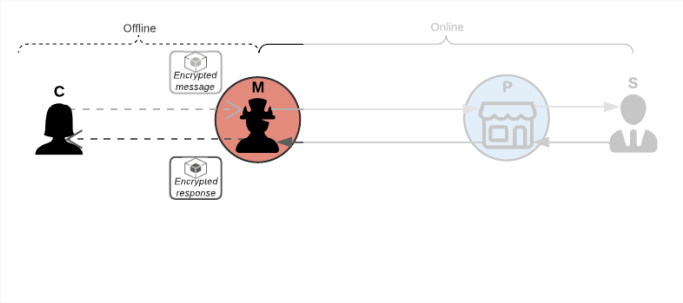
- ***Client*** (***C***) is offline and wants to send a message to a ***Server*** (***S***) that is online
- A ***Data Mule*** (***M***) retrieves (offline) **C**'s payload and brings it to a ***Proxy*** (***P***).
- ***P*** in turn forwards (online) the message to ***S***.

InDaMul



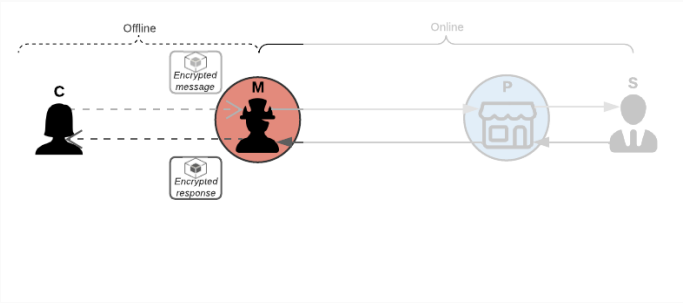
- ***Client*** (***C***) is offline and wants to send a message to a ***Server*** (***S***) that is online
- A ***Data Mule*** (***M***) retrieves (offline) ***C***'s payload and brings it to a ***Proxy*** (***P***).
- ***P*** in turn forwards (online) the message to ***S***.
- Then, a message can be returned from ***S*** to ***C*** in the opposite way.

Client to Mule



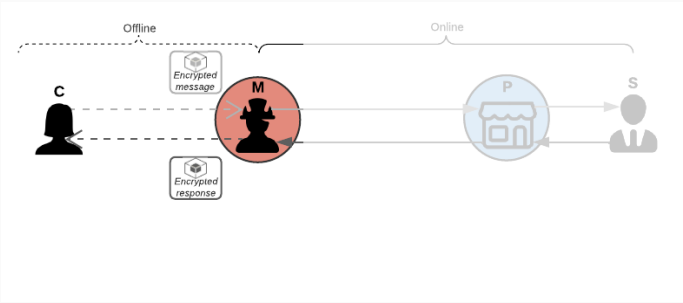
- **C** waits a Mule to pick up its payload p_C encrypted with a key x.

Client to Mule



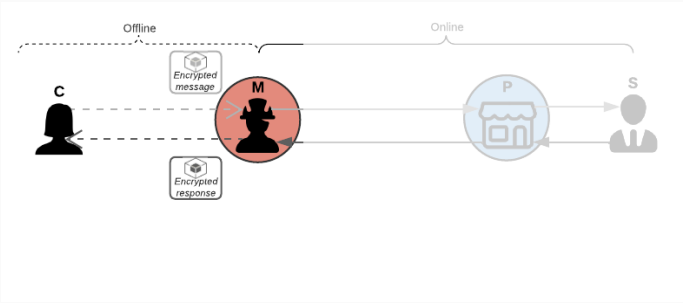
- **C** waits a Mule to pick up its payload p_C encrypted with a key x.
- A Mule **M1** passing nearby accepts to carry the message.

Client to Mule



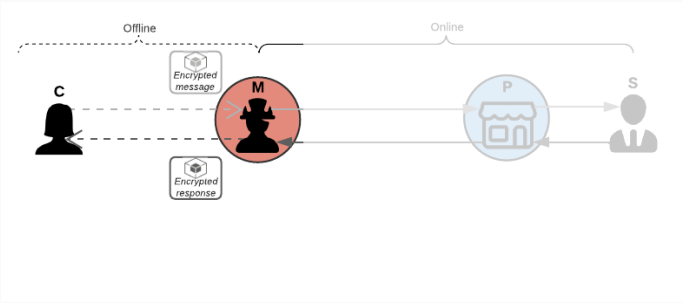
- **C** waits a Mule to pick up its payload p_C encrypted with a key x.
- A Mule **M1** passing nearby accepts to carry the message.
- **C** transmits to **M1** the payload p_C and also:

Client to Mule



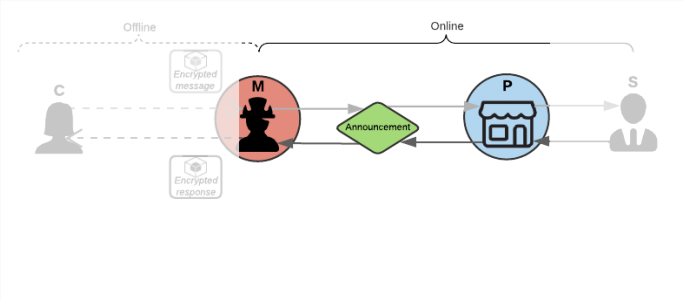
- **C** waits a Mule to pick up its payload p_C encrypted with a key x.
- A Mule **M1** passing nearby accepts to carry the message.
- **C** transmits to **M1** the payload p_C and also:
  - a $balance_{M1}$ object used to update the state channel between **C** and **M1**.

Client to Mule



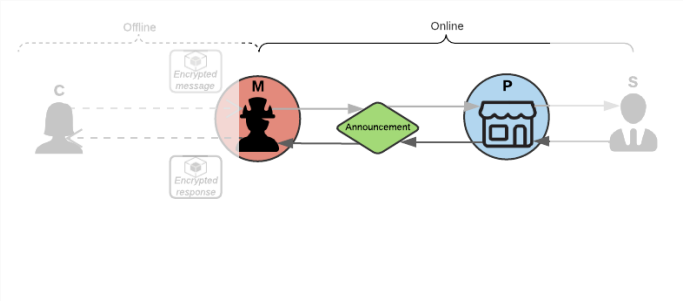
- **C** waits a Mule to pick up its payload p_C encrypted with a key x.
- A Mule **M1** passing nearby accepts to carry the message.
- **C** transmits to **M1** the payload p_C and also:
  - a $balance_{M1}$ object used to update the state channel between **C** and **M1**.
  - $tender_C$ containing: an immutable URI_p of p_C; *offer* to a P; id_x of the encryption key.

Mule to Proxy [1/3]



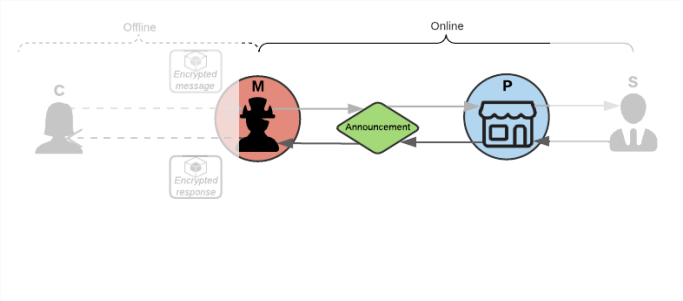
- When **M1** becomes online it publishes the *tender* in an **Announcement Service**, in order to reach an audience of different Proxies.

Mule to Proxy [1/3]



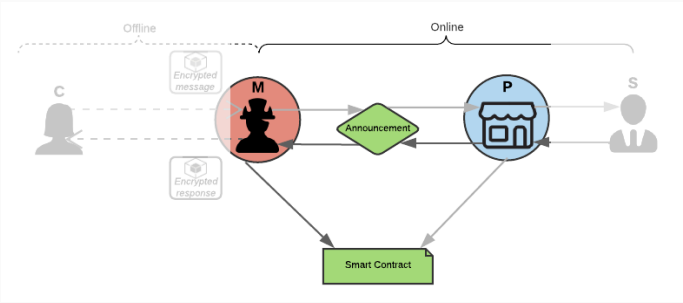
- When **M1** becomes online it publishes the *tender* in an **Announcement Service**, in order to reach an audience of different Proxies.
- While announcing the tender, **M1** also uploads p_C to a **DFS**.

Mule to Proxy [1/3]



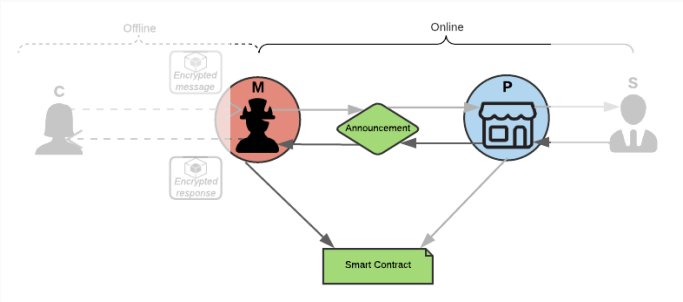
- When **M1** becomes online it publishes the *tender* in an **Announcement Service**, in order to reach an audience of different Proxies.
- While announcing the tender, **M1** also uploads p_C to a **DFS**.
- A Proxy **P**, which decides to take charge of p_C, downloads payload and *tender*.

Mule to Proxy [2/3]



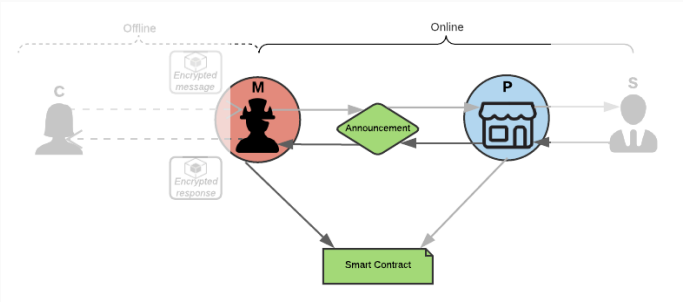
- A Smart Contract named ***InDaMul*** executes the majority of the protocol tasks.

Mule to Proxy [2/3]



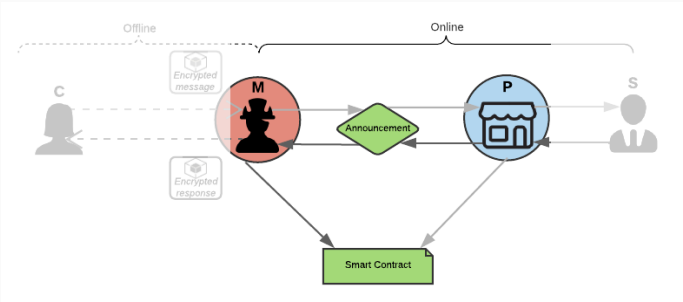
- A Smart Contract named ***InDaMul*** executes the majority of the protocol tasks.
- **P** simply invokes a method in ***InDaMul*** using the *tender* object as input.

Mule to Proxy [2/3]



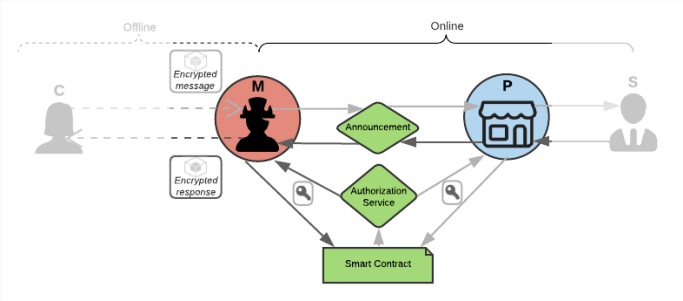
- A Smart Contract named ***InDaMul*** executes the majority of the protocol tasks.
- **P** simply invokes a method in ***InDaMul*** using the *tender* object as input.
- ***submitTender*** method automatically checks the validity of the signatures found in the data provided by **M1**, i.e. the *tender*, and then binds **P**'s address with id_x.

Mule to Proxy [2/3]



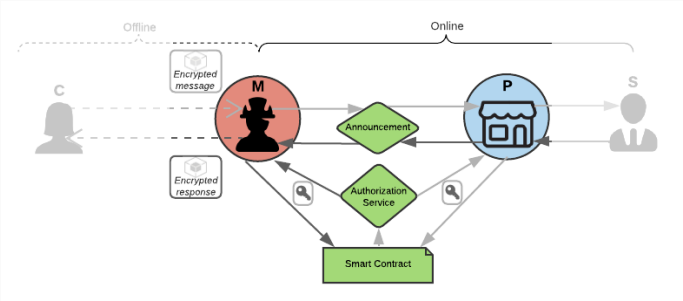
- A Smart Contract named ***InDaMul*** executes the majority of the protocol tasks.
- **P** simply invokes a method in ***InDaMul*** using the *tender* object as input.
- ***submitTender*** method automatically checks the validity of the signatures found in the data provided by **M1**, i.e. the *tender*, and then binds **P**'s address with id_x.
- This makes **P** eligible to get access to the key identified by id_x.

Mule to Proxy [3/3]



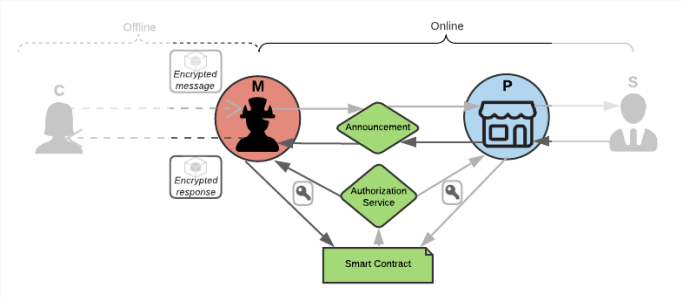
- **P** sends request to the decentralized **Authorization Service** for accessing the key *x*

Mule to Proxy [3/3]



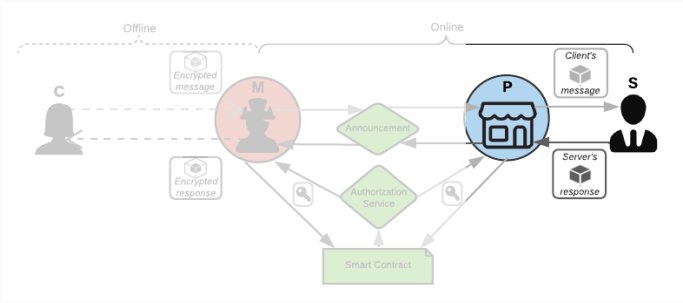
- **P** sends request to the decentralized **Authorization Service** for accessing the key *x*
- This consists of a subset of DLT nodes maintaining shares of the key *x* using the Secret Sharing cryptographic technique.

Mule to Proxy [3/3]



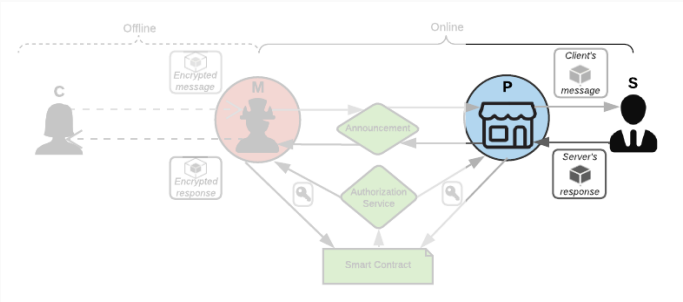
- **P** sends request to the decentralized **Authorization Service** for accessing the key *x*
- This consists of a subset of DLT nodes maintaining shares of the key *x* using the Secret Sharing cryptographic technique.
- Each authorization node releases a share of *x* to **P** after checking the ledger

Proxy to Server



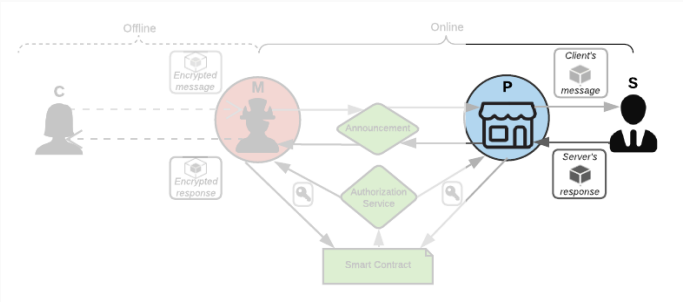
- **P** aggregates shares to decrypt the payload p_C and send it to **S**.

Proxy to Server



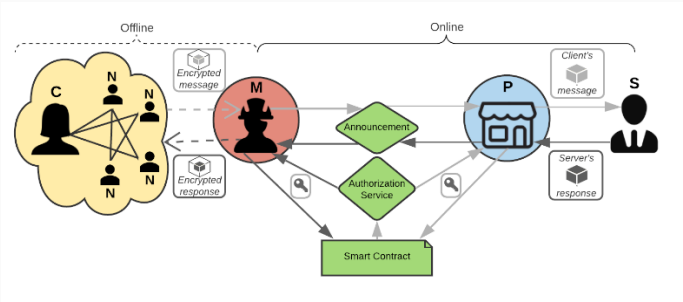
- **P** aggregates shares to decrypt the payload p_C and send it to **S**.
- *submitTender* automatically locked an amount of tokens in favor of **P** until a response reaches **C** through another Mule **M2**.

Proxy to Server



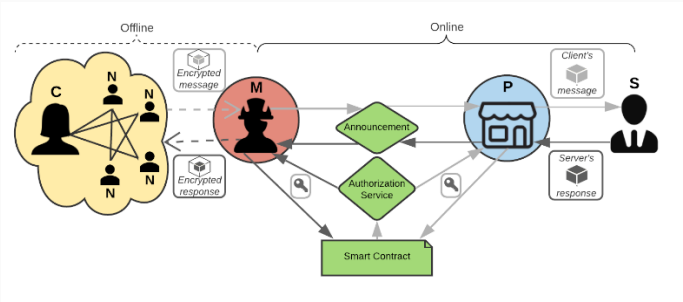
- **P** aggregates shares to decrypt the payload p_C and send it to **S**.
- *submitTender* automatically locked an amount of tokens in favor of **P** until a response reaches **C** through another Mule **M2**.
- *balance* objects are locked in the *InDaMul* smart contract for **M1** and **M2** until some conditions are verified by the data given in input.

“The Island”



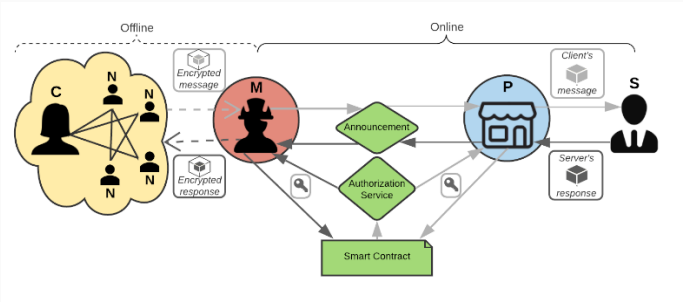
- When **C** is not in the action range of any **Mule**, a network can be set up between **C**’s physical **Neighbors** (**N**) -> **the “Island”**.

“The Island”



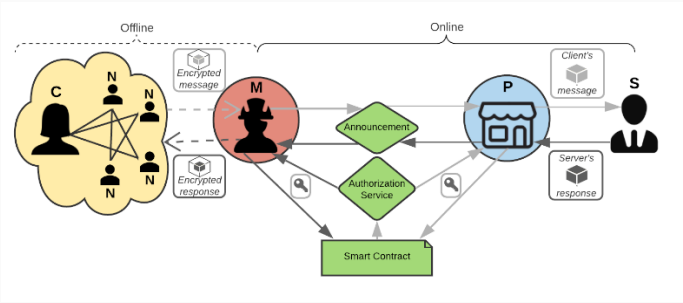
- When **C** is not in the action range of any **Mule**, a network can be set up between **C**’s physical **Neighbors** (**N**) -> **the “Island”**.
- One or more **Target Neighbors** (**TN**), must be reached by a **Mule** and act as relays.

“The Island”



- When **C** is not in the action range of any **Mule**, a network can be set up between **C**’s physical **Neighbors** (**N**) -> **the “Island”**.
- One or more **Target Neighbors** (**TN**), must be reached by a **Mule** and act as relays.
- Clients that cannot interact directly with a *TN* have to find a path within the Island.

“The Island”



- When **C** is not in the action range of any **Mule**, a network can be set up between **C**’s physical **Neighbors** (**N**) -> **the “Island”**.
- One or more **Target Neighbors** (**TN**), must be reached by a **Mule** and act as relays.
- Clients that cannot interact directly with a *TN* have to find a path within the Island.
- To incentivize Neighbors to relay messages a State Channel Network is used

Conclusions and Future Work

- Implementations in **Ethereum** ->
  submitTender method in the *InDaMul* contract -> high gas usage, i.e. $\sim 246k$.

Conclusions and Future Work

- Implementations in **Ethereum** ->
  submitTender method in the *InDaMul* contract -> high gas usage, i.e. $\sim 246k$.
- -> **Sidechain** (e.g. Polygon)

Conclusions and Future Work

- Implementations in **Ethereum** ->
  submitTender method in the *InDaMul* contract -> high gas usage, i.e. $\sim 246k$.
- -> **Sidechain** (e.g. Polygon)
- Future work -> focus on the **Mules mobility** since other delays, by comparison, are negligible.

Conclusions and Future Work

- Implementations in **Ethereum** ->
  submitTender method in the *InDaMul* contract -> high gas usage, i.e. $\sim 246k$.
- -> **Sidechain** (e.g. Polygon)
- Future work -> focus on the **Mules mobility** since other delays, by comparison, are negligible.
- **Smart-village-like scenario** where buses and couriers act as Data Mule.

Conclusions and Future Work

- Implementations in **Ethereum** ->
  submitTender method in the *InDaMul* contract -> high gas usage, i.e. $\sim 246k$.
- -> **Sidechain** (e.g. Polygon)
- Future work -> focus on the **Mules mobility** since other delays, by comparison, are negligible.
- **Smart-village-like scenario** where buses and couriers act as Data Mule.
- Simulations performed with the **LUNES** agent-based simulator.