UNIVERSITY
OF AMSTERDAM

# Trustworthy Cloud Service Level Agreement Enforcement with Blockchain based Smart Contract

Huan Zhou, Cees de Laat and **Zhiming Zhao**

# Outline

UNIVERSITY OF AMSTERDAM

1. Background and Challenges of Cloud SLA / Blockchain
2. Related Work
3. Model Design and System Architecture
4. Key Techniques to Ensure Trustworthiness
   - Unbiased Random Sortition Algorithm
   - Payoff Function Design and Nash Equilibrium
5. Conclusion
6. Further Work

# What is Cloud SLA?

UNIVERSITY OF AMSTERDAM

- Traditionally, SLA (Service Level Agreement) is a *business* concept which defines the contractual financial agreements between the roles who are engaging in the business activity.
- Cloud SLA is an agreement between the cloud *customer* and *provider* on the quality of the cloud service.

For instance

Cloud provider, Amazon Elastic Compute Cloud (EC2), claims that the availability of its data center is no less than **99%**. If this number is not achieved, it will pay back **30%** credits to its customer as compensation.

# What are the challenges of Cloud SLA?

UNIVERSITY OF AMSTERDAM

- **Manual Verification:** There lacks of an automatic mechanism to enforce the agreement, especially the automated compensation after SLA violation. The current process involves a lot of manual work such as doing the verification through emails before compensation.

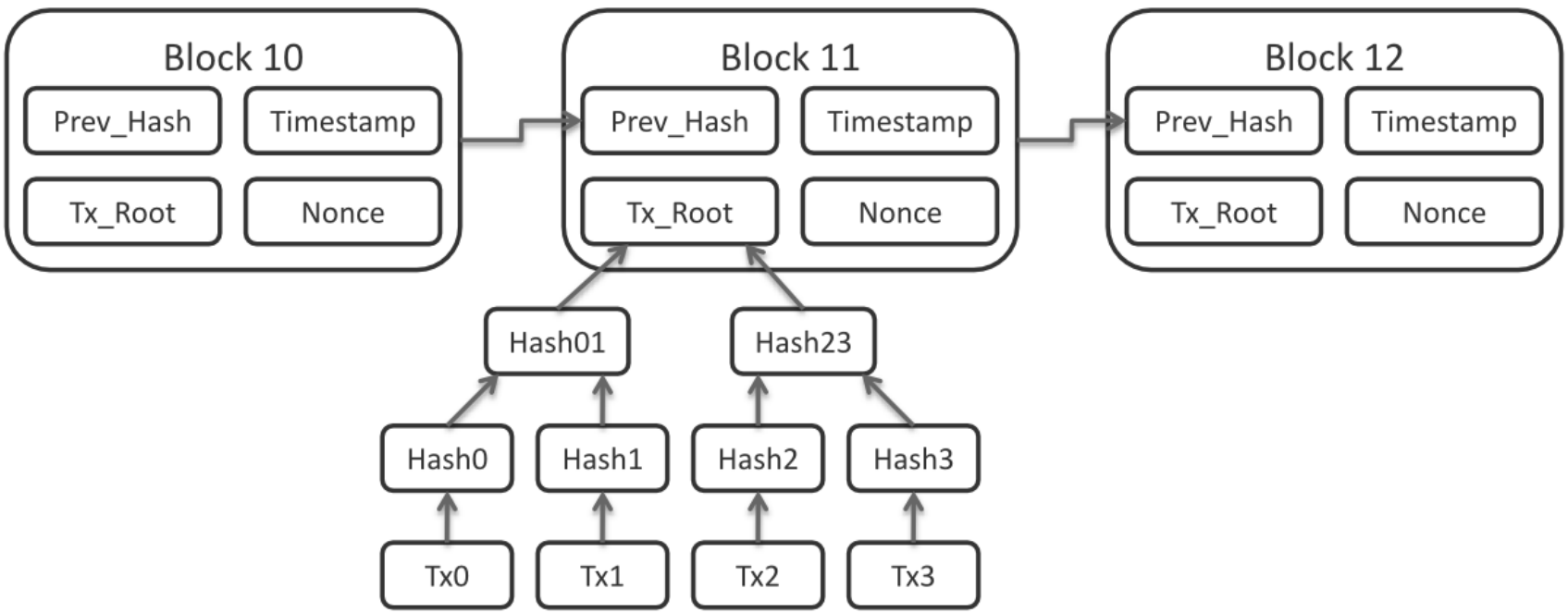
- **Fairness:** The provider has the right to verify the violation and decide whether to compensate the customer. On the contrary, the customer has little space to negotiate about the price or the amount of compensation.

- **Violation Proof:** It is hard for the customer to prove and convince the provider that the violation has really occurred.

# What is Blockchain?

UNIVERSITY OF AMSTERDAM

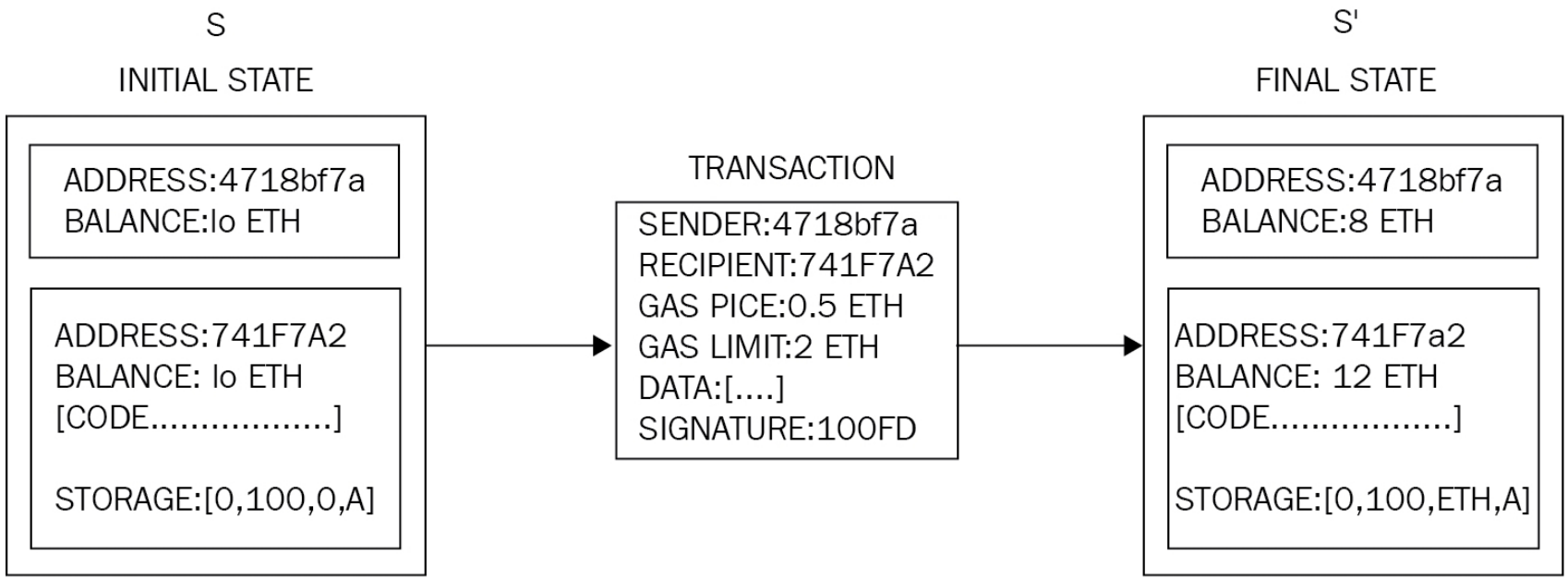
- Blockchain is a technique, which makes every participant having consensus on a decentralized ledger through PoW (proof of work) or PoS (proof of stake), etc.
- Bitcoin is the first generation application of blockchain, from 2009.
  - max 7 tx/s
  - about 10 mins for new block



# What is Ethereum and Smart Contract?

UNIVERSITY OF AMSTERDAM

- Ethereum is the second generation blockchain, from 2015.
  - max around 20 tx/s
  - about 15 secondes for new block
- It proposes EVM (Ethereum Virtual Machine), which is a set of byte values to represent a virtual machine state.
- The transaction is a program to change the EVM state.
- The program running on the blockchain is named as *Smart Contract*.



# What is the challenge of Blockchain?

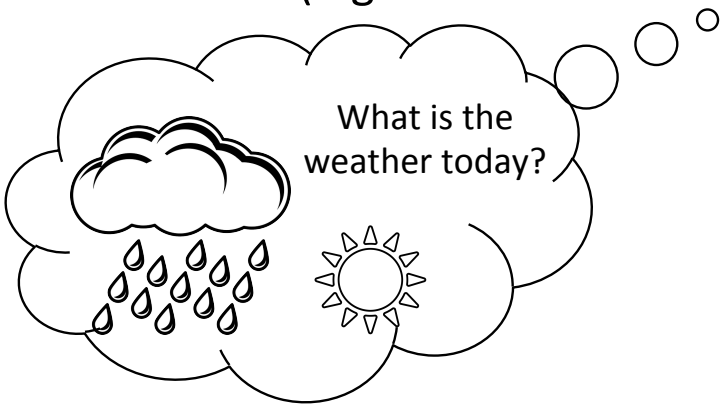
UNIVERSITY OF AMSTERDAM


***Off-chain*** Events
(e.g. service violation)

***Who*** provides the information and data?

Is it ***trustworthy***?

***On-chain*** Transactions
(e.g. violation detection)




In the context of Cloud SLA, who can convince both, the provider and customer, that the service violation really happens? How?

# Related Work

UNIVERSITY OF AMSTERDAM

About "oracle" of blockchain:

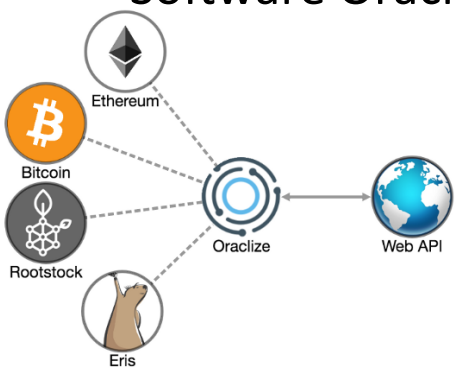
***Who?***
***Oracle***: Perform as "Data Carrier"

***Off-chain*** Events → ***On-chain*** Transactions

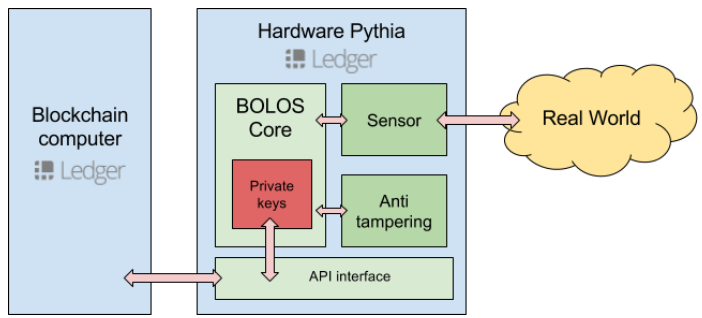
Is it ***trustworthy***?

## Third trusted party

### Software Oracle

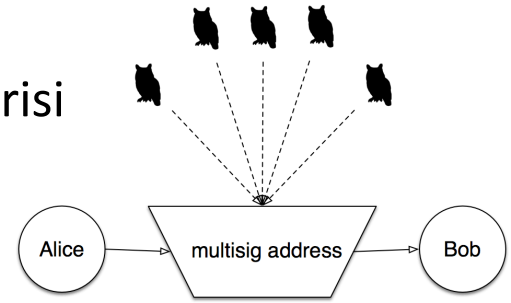



### Hardware Oracle




## Distributed oracles

Orisi




For participants:

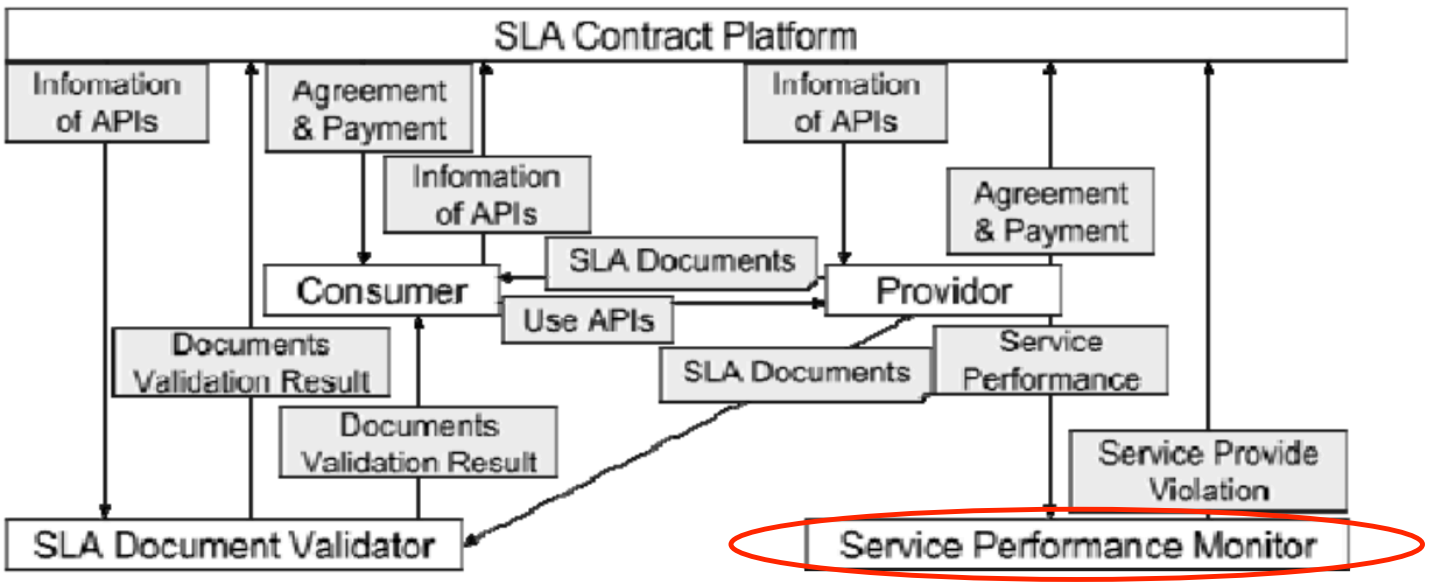
- Require them independent and trustworthy;
- No incentive;
- Consensus issue;

# Related Work

UNIVERSITY OF AMSTERDAM

About SLA automation:

- Hiroki [1] leverages Ethereum to automate the SLA lifecycle enforcement on the blockchain.
- They introduced "*Service Performance Monitor*" to be responsible for the violation management and report.
- It is not discussed whether those violation reports are trustworthy.

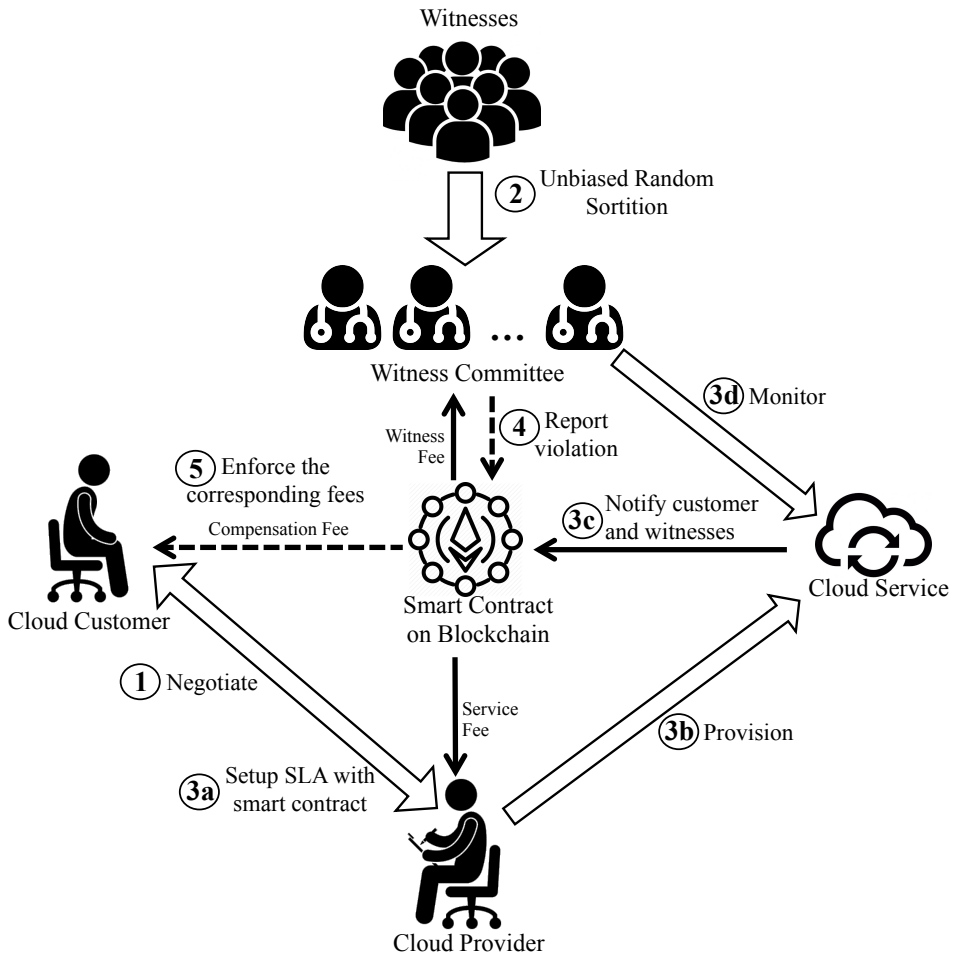



[1] H. Nakashima and M. Aoyama, "An automation method of SLA contract of web APIs and its platform based on blockchain concept," in *Cognitive Computing (ICCC), 2017 IEEE International Conference on*. 2017, pp. 32-39.

# Model Design and System Architecture

UNIVERSITY OF AMSTERDAM

- Introducing a new role to the traditional cloud SLA, "*Witness*".
- Its responsibility is service monitoring and violation report.
- Any blockchain user can register as a witness, whose motivation is to earn revenue.



# Key Techniques to Ensure Trustworthiness
## (1/2) Unbiased Random Sortition Algorithm

UNIVERSITY OF AMSTERDAM

---

**Input:**

Registered witness set, $RW$;
Required number, $N$, of members in witness committee;
Random number, $rand_p$, given by the provider $p$;
Random number, $rand_c$, given by the customer $c$.

**Output:**

Selected witness set, $SW$, to form a committee.

1: $rand_{hash} \leftarrow Hash(rand_p + rand_c)$
2: $j \leftarrow 0$
3: $i \leftarrow 1$
4: $SW \leftarrow \emptyset$
5: **for all** $w_i$ such that $w_i \in RW$ **do**
6: **if** $addr(w_i) > rand_{hash}$ && $w_i$ is online **then**
7: add $w_i \Rightarrow SW$
8: $j++$
9: **end if**
10: **if** j == N **then**
11: break;
12: **end if**
13: $i++$
14: $i \leftarrow (i \bmod (\|RW\| + 1)) + 1$
15: **end for**
16: return $SW$

---

# Key Techniques to Ensure Trustworthiness

## (1/2) Unbiased Random Sortition Algorithm

improved version: **randomness** from **blockchain itself**

UNIVERSITY OF AMSTERDAM

**Input:**

Registered witness set, $RW$;
The the size of the list, $len(RW)$;
The number of online witnesses, $oc$;
$N$ members in witness committee;
The hash value, $B_i^{hash}$, of the $i^{th}$ block;
The block index, $b$, at the sortition request;
Current block index, $Id$;
Following sequential, $K_s$, blocks;
$K_c$ blocks for confirmation;
The addresses of the provider, $address(p)$;
The addresses of the customer, $address(c)$.
(Remarks: recommended values are $K_s = 10$, $K_c = 12$)

**Output:**

Selected witness committee, $SW$.

1: assert($Id > b + K_s + K_c$)
2: assert($oc >= 10 * N$)
3: seed $\leftarrow 0$
4: **for all** $i = 1$ ; $i <= K_s$ ; $i++$ **do**
5: seed$+= B_{b+1+i}^{hash}$
6: **end for**
7: $SW \leftarrow \emptyset$
8: $j \leftarrow 0$
9: **while** $j < N$ **do**
10: $index \leftarrow$ seed%$len(RW)$
11: **if** $state(RW[index]) == Online$ && $reputation(RW[index]) > 0$ && $RW[index]! = address(c)$ && $RW[index]! = address(p)$ **then**
12: $state(RW[index]) \leftarrow Candidate$
13: $oc--$
14: Add $RW[index] \Rightarrow SW$
15: $j++$
16: **end if**
17: seed $\leftarrow hash$(seed)
18: **end while**
19: return $SW$

# Key Techniques to Ensure Trustworthiness
## (2/2) Payoff Function Design and Nash Equilibrium

UNIVERSITY OF AMSTERDAM

**Witness Game**: It is a *n-player* game represented as a triple ($SW$, $\Sigma$, $\Pi$), where

- $SW = \{w_1, w_2, ..., w_n\}$ is a set of $n$ players. Each player is a witness and they form the witness committee.
- $\Sigma = \Sigma_1 \times \Sigma_2 \times ... \times \Sigma_n$ is a set of strategy profiles, where $\Sigma_k$ is a set of actions for the witness $w_k$. A strategy profile is therefore a vector, $\sigma = (\sigma_1, \sigma_2, ..., \sigma_n)$, where $\sigma_k$ is a possible action of $\Sigma_k$, $(k = 1, 2, .., n)$.
- $\Pi = \{\pi_1, \pi_2, ..., \pi_n\}$ is a set of payoff functions, where $\pi_k : \Pi \rightarrow R$ is the payoff function determining the revenue for witness $w_k$, $(k = 1, 2, .., n)$. $R$ is the corresponding revenue.
- To be specific, $\Sigma_k = \{\sigma_k^{(r)}, \sigma_k^{(s)}\}$:
  - $\sigma_k^{(r)}$ means *R*eport the service violation to the smart contract;
  - $\sigma_k^{(s)}$ means do not report and keep *S*ilence to the smart contract.

$\Rightarrow$ This is the basic type of **Strategic Form Game with Complete Information** in game theory.

In a $N$ witnesses game, one of the strategy profile is, $\sigma = (\sigma_1^*, \sigma_2^*, ..., \sigma_N^*)$, where $\sigma_k^*$ is a specific action adopted by witness $w_k$. Define the witness set choosing a specific action as follows.

- $W_{report}$: $\forall w_k \in W_{report}$, $\sigma_k^* = \sigma_k^{(r)}$;
- $W_{silence}$: $\forall w_k \in W_{silence}$, $\sigma_k^* = \sigma_k^{(s)}$.

# Key Techniques to Ensure Trustworthiness
## (2/2) Payoff Function Design and Nash Equilibrium

UNIVERSITY OF AMSTERDAM

**Violation Confirmation**: Based on the result of a strategy profile in a $N$ witnesses game, the service violation is confirmed, $SLA_{status} = Violated$, only when $\|W_{report}\| \geq M$, where $1 < N/2 < M < N, N, M \in \mathbb{N}$. Otherwise, it is treated as there is no violation happened, $SLA_{status} = Completed$.

**Payoff Functions**

- When $SLA_{status} = Violated$:
  $\forall w_k \in W_{report}, \pi_k = 10$;
  $\forall w_k \in W_{silence}, \pi_k = 0$.
- When $SLA_{status} = Completed$:
  $\forall w_k \in W_{report}, \pi_k = -1$;
  $\forall w_k \in W_{silence}, \pi_k = 1$

# Key Techniques to Ensure Trustworthiness
## (2/2) Payoff Function Design and Nash Equilibrium

UNIVERSITY OF AMSTERDAM

**Witness $w_k$'s best response:** In order to maximize its revenue, $w_k$'s best response to a strategy profile $\sigma_{-k}$, i.e. $\{\sigma_1, \sigma_2, ..., \sigma_{k-1}, \sigma_{k+1}, ..., \sigma_n\}$, is a strategy $\sigma_k^* \in \Sigma_k$, such that $\pi_i(\sigma_k^*, \sigma_{-k}) \geq \pi_k(\sigma_k, \sigma_{-k})$ for $\forall \sigma_k \in \Sigma_k (k = 1, 2, ..., n)$.

**Nash equilibrium:** It is a specific strategy profile $\sigma^* = (\sigma_k^*, \sigma_{-k}^*)$, if for each witness $w_k$, $\sigma_k^*$ is a best response to $\sigma_{-k}^*$, i.e., $\pi_i(\sigma_k^*, \sigma_{-k}^*) \geq \pi_k(\sigma_k, \sigma_{-k}^*)$ for $\forall \sigma_k \in \Sigma_k (k = 1, 2, ..., n)$.

### Theorem

*In the witness game, the only two Nash equilibrium points are following strategy profiles:*

- $\sigma^* = (\sigma_1^*, \sigma_2^*, ..., \sigma_n^*)$, *of which for* $\forall w_k$, $\sigma_k^* = \sigma_k^{(r)}$;
- $\sigma^* = (\sigma_1^*, \sigma_2^*, ..., \sigma_n^*)$, *of which for* $\forall w_k$, $\sigma_k^* = \sigma_k^{(s)}$.

# Key Techniques to Ensure Trustworthiness
## (2/2) Payoff Function Design and Nash Equilibrium

UNIVERSITY OF AMSTERDAM

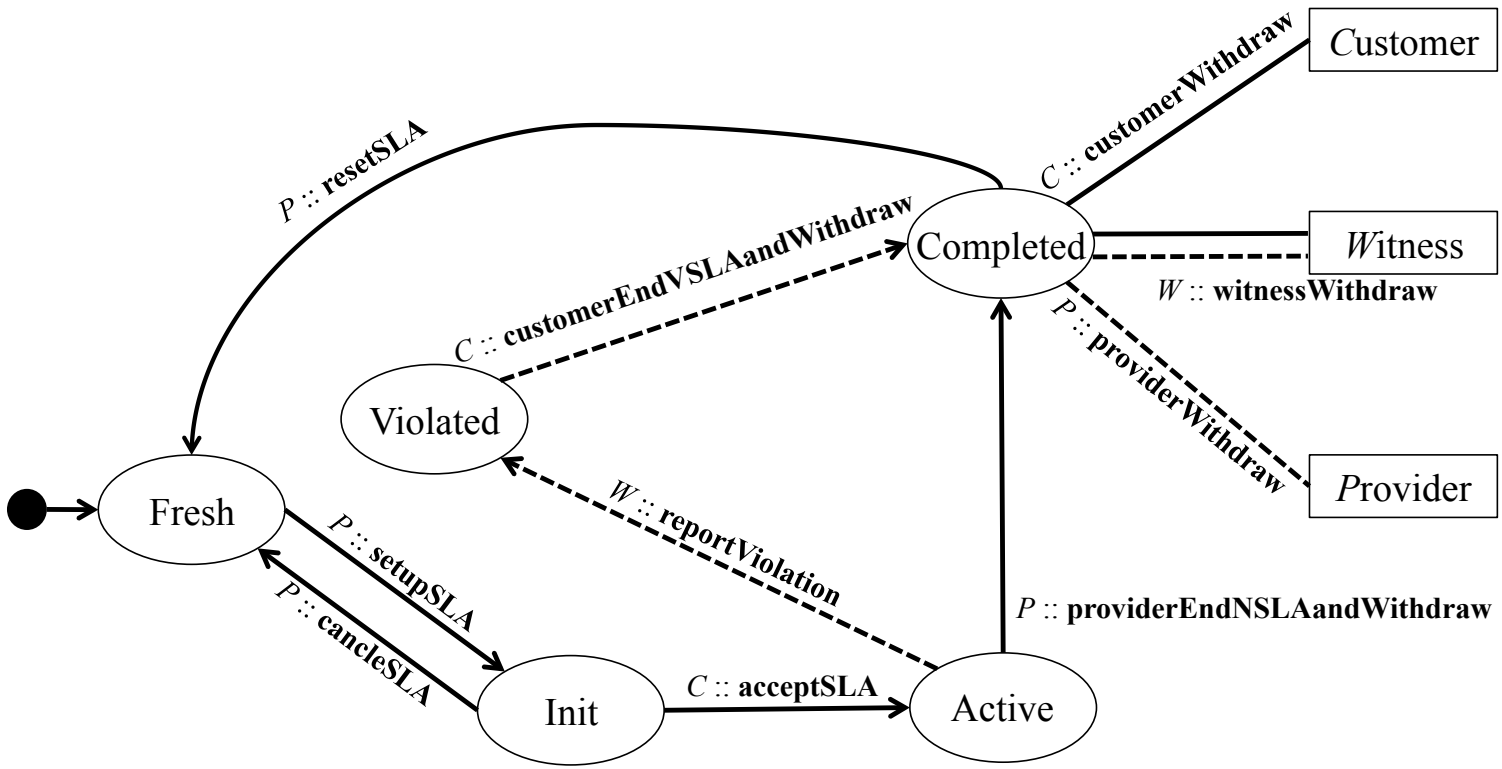
- Take the example of a three-witness game ($N = 3, M = 2$).
- Their payoffs according to different actions are shown in the table.

| $w_1$ : *Alice* | $w_3$ : *Candy* | | | |
|---|---|---|---|---|
| | $\sigma_3^{(r)}$: *Report* | | $\sigma_3^{(s)}$: *Silence* | |
| | $w_2$ : *Bob* | | $w_2$ : *Bob* | |
| | $\sigma_2^{(r)}$: *Report* | $\sigma_2^{(s)}$: *Silence* | $\sigma_2^{(r)}$: *Report* | $\sigma_2^{(s)}$: *Silence* |
| $\sigma_1^{(r)}$: *Report* | (10, 10, 10) | (10, 0, 10) | (10, 10, 0) | (-1, 1, 1) |
| $\sigma_1^{(s)}$: *Silence* | (0, 10, 10) | (1, 1, -1) | (1, -1, 1) | (1, 1, 1) |

# Implementation Details
## A Specific SLA Smart Contract

UNIVERSITY OF AMSTERDAM

- This is the smart contract to enforce a specific SLA lifecycle.
- The figure below shows how interfaces are implemented in the smart contract[2] to realize the **SLA state transition**.




[2] https://github.com/zh9314/SmartContract4SLA

# Conclusion

UNIVERSITY OF AMSTERDAM

- A **witness model** is proposed for cloud SLA enforcement and specially design the **payoff function** for each witness;
- The **game theory** is leveraged to analysis that the witness **has to** offer honest monitoring service, in order to maximize its own revenue;
- A **prototype system** is fully implemented using **smart contracts** of Ethereum to realize the witness model, including witnesses management.

$\Longrightarrow$ To the best of our knowledge, this is the first work that applies **economic principles** to achieving **trustworthy consensus** among oracles.

# Future Work

UNIVERSITY OF AMSTERDAM

- **On-chain** work
  - Further optimize the interface implementation to reduce the gas consumption;
  - Enrich the functionalities of the smart contract;
  - Consider some more application scenarios, not only cloud SLA.

- **Off-chain** work
  - User-friendly tools are required for each role in the system to perform their corresponding interactions with the smart contract;
  - Combine our previous work, CloudsStorm, to construct the witness ecos-ystem.

# Acknowledgment

This work was supported by the European Union's Horizon 2020 research and innovation program under grant agreements, No. 643963 (**SWITCH** project), No. 654182 (**ENVRIPLUS** project) and No. 676247 (**VRE4EIC** project).