Enhancing Trust of Supply Chain Using Blockchain Platform with Robust Data Model and Verification Mechanisms

KAIST

Byungsoo Oh, Tae Joon Jun, Wondeuk Yoon,

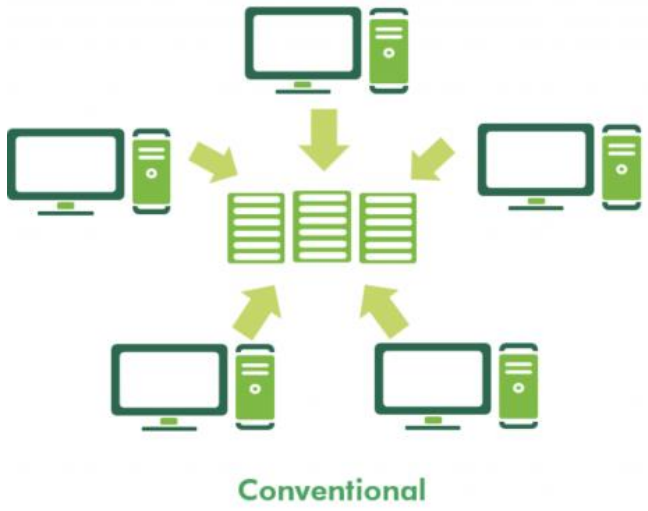
Yunho Lee, Sangtae Kim, Daeyoung Kim

Fipronil egg contamination (2017)

- Took long to identify where and how contamination started and spread out
- Raised questions about **visibility** of supply chains

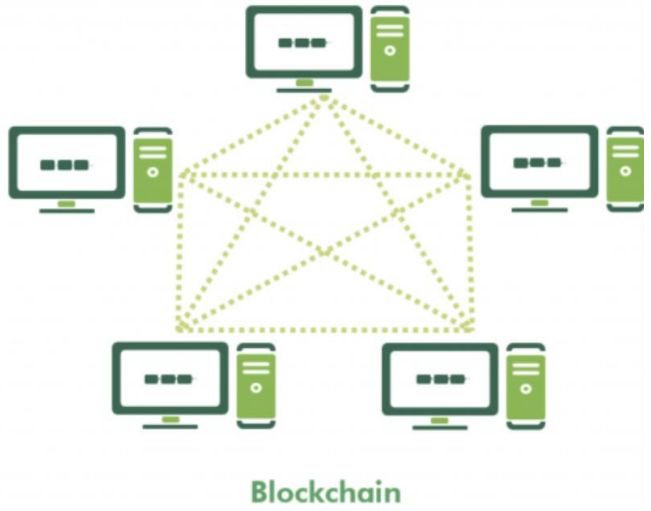
Rise of blockchain-based solutions

Independent & Fragmented systems



- **Fragile** against data tampering
- Requires reconciliation of data → track & trace: **complex & time-consuming**

Shared & Distributed systems

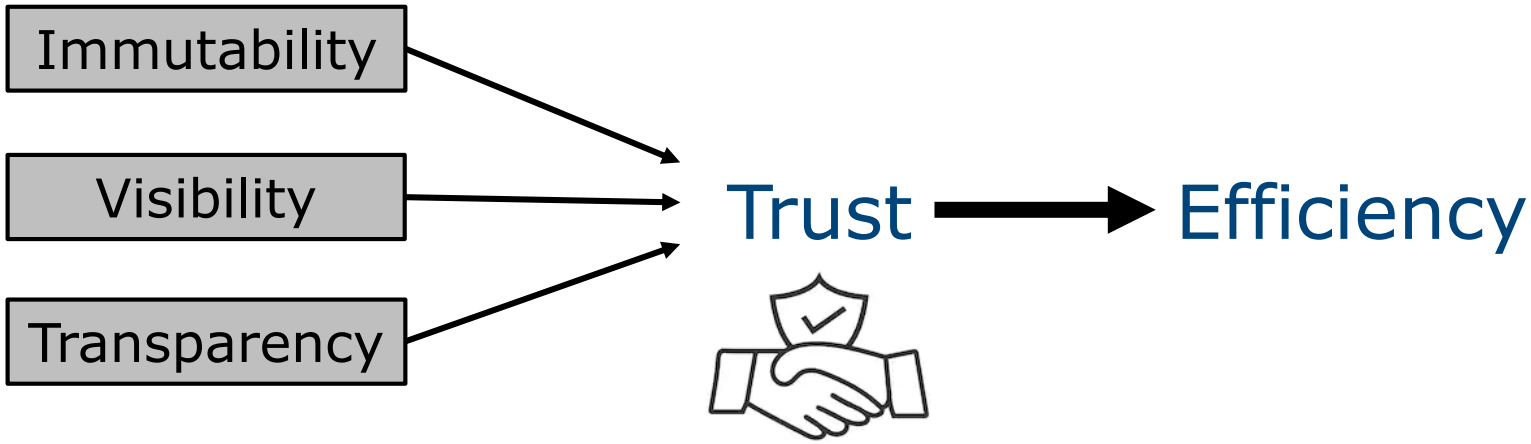


- **Robust** against data tampering (cryptography-based verification)
- Distributed consensus & smart contract-based automation → track & trace: **simple & fast**

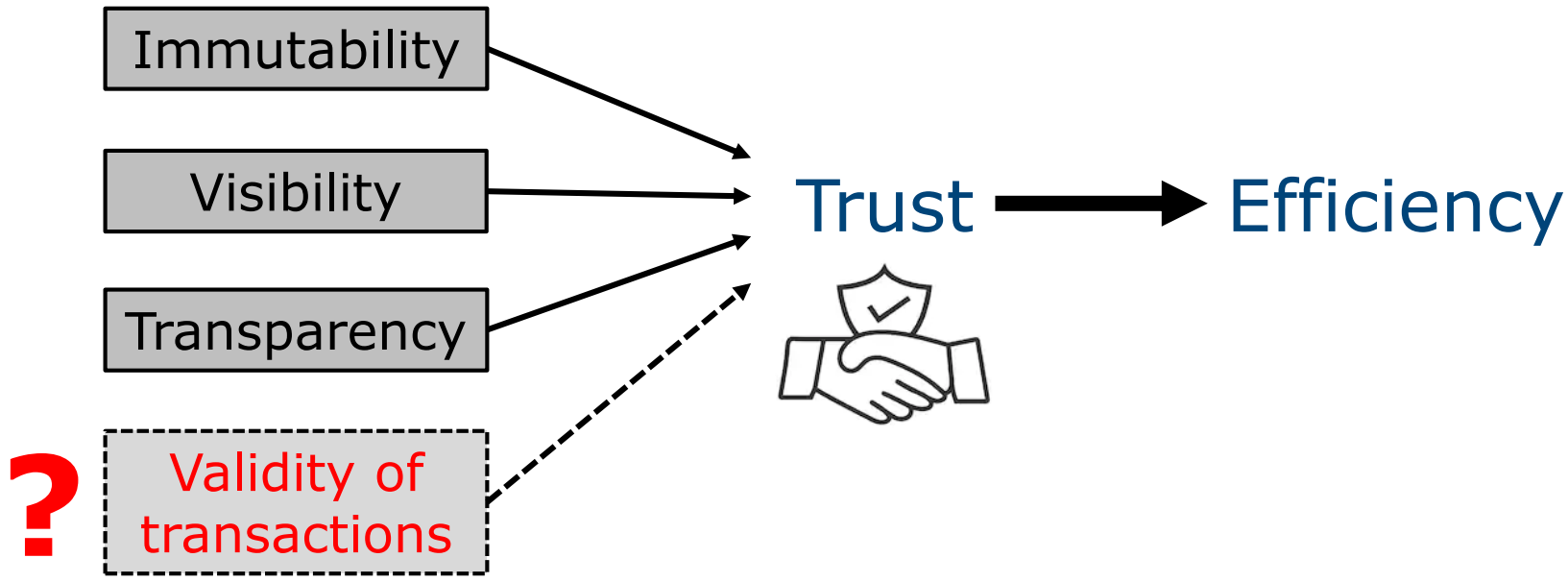
Rise of blockchain-based solutions

Conventional SCM system		Blockchain-based SCM system
• Independent, fragmented ledgers	VS	• Shared ledgers
• Limited visibility		• Full visibility
• Low availability		• High availability
• Fragile to data tampering		• Resilient to data tampering
• Costly traceability		• Cost-efficient traceability

Trust model of blockchain

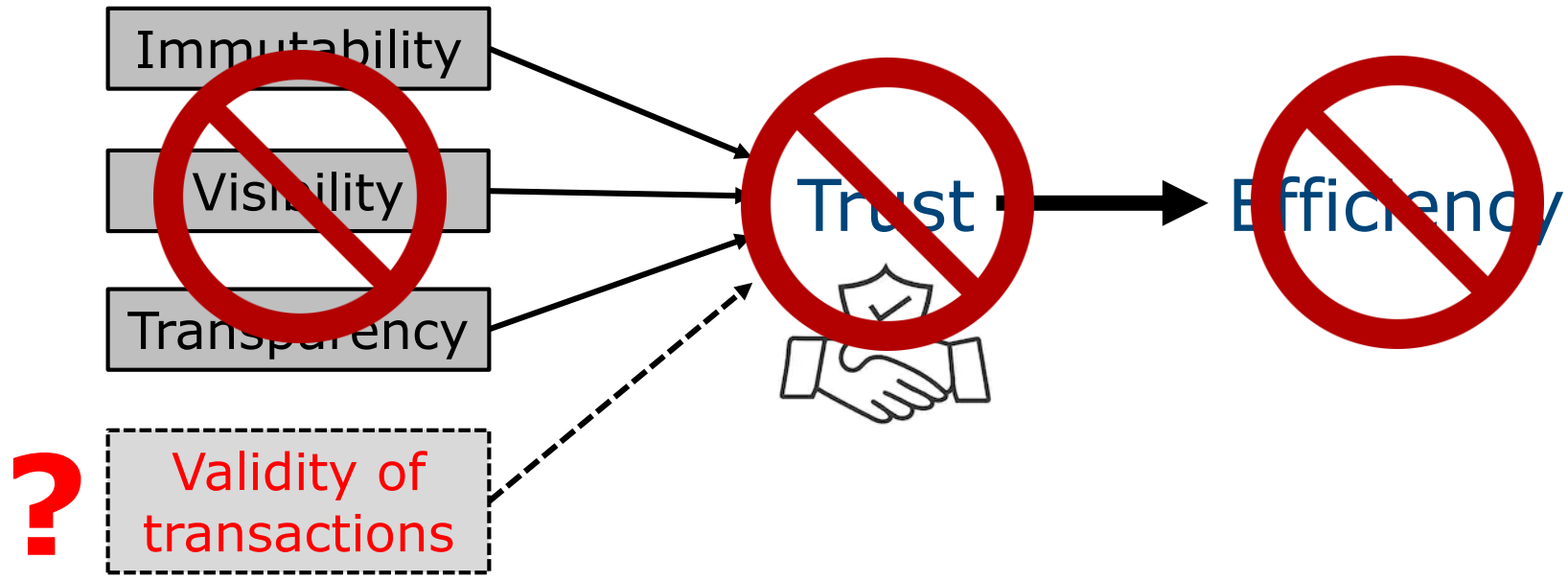


What's often neglected



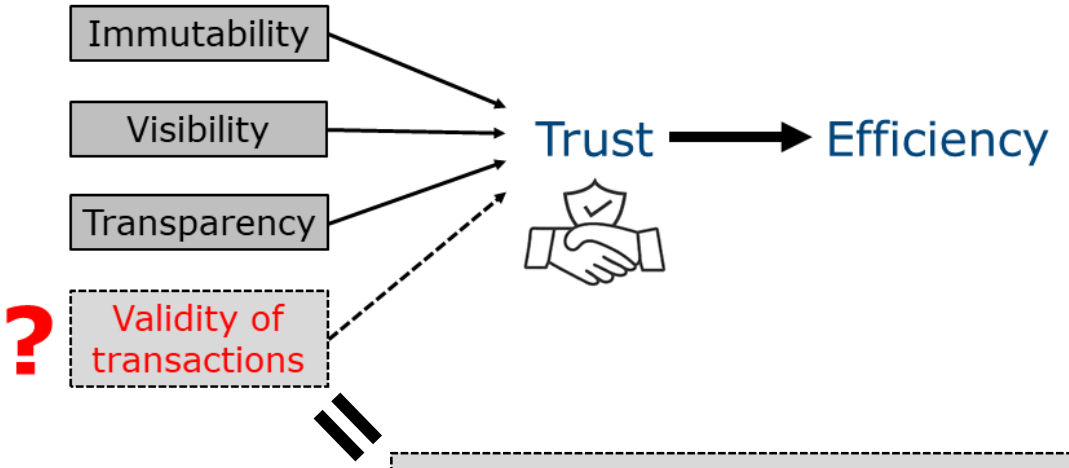
Question: What if **wrong data** is committed in blockchain?

Without proper data verification



Then, almost all benefits of blockchain become **meaningless**

Validity of transactions?



(1) Programming language aspects of smart contracts
- Actively studied (e.g. symbolic model checking – NDSS'18)

(2) Transactions aligned with **business context**
- No work up to date

Especially important for supply chain management (SCM) to be **robust against anomalous actions**

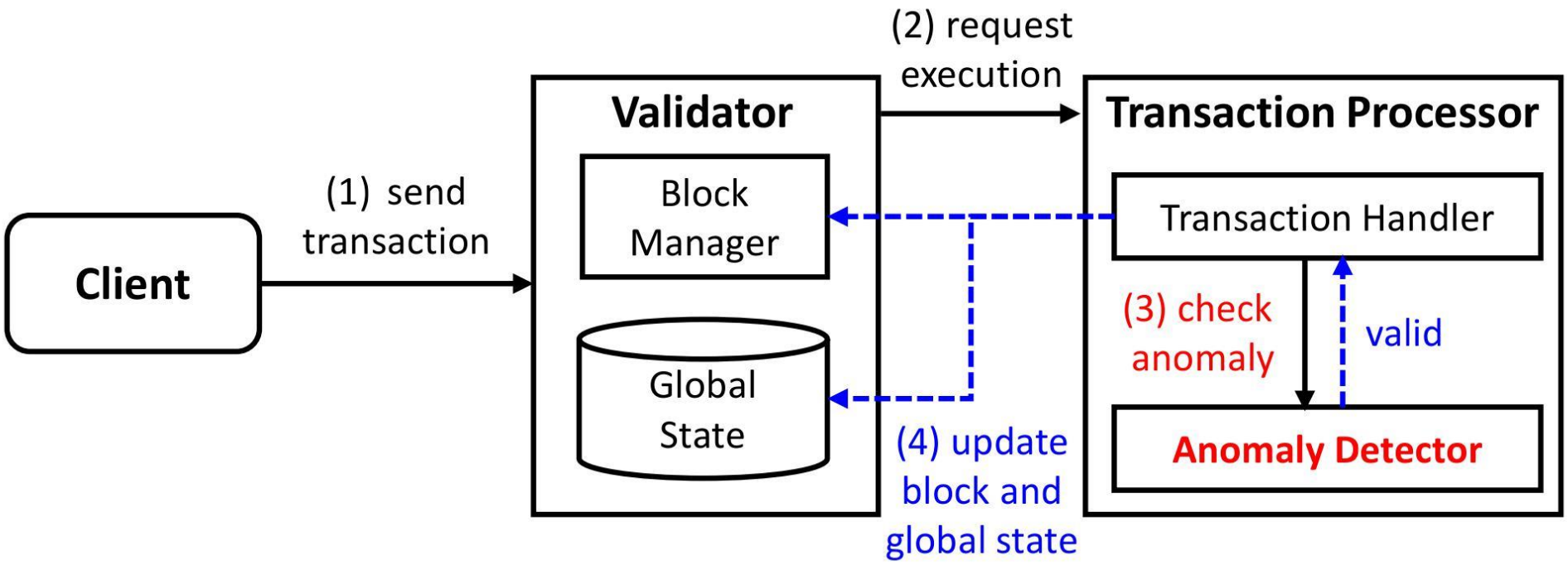
Goal

- Anomaly detection: Filtering of transactions that are inconsistent with existing business context
- Goal: Propose a blockchain platform for SCM that has anomaly detection layer to filter anomalous transactions
  - Robust data model
  - Verification mechanisms

Outline

- Motivation and goals
- System overview
- Data model
- Verification of business logic
- Implementation and evaluation

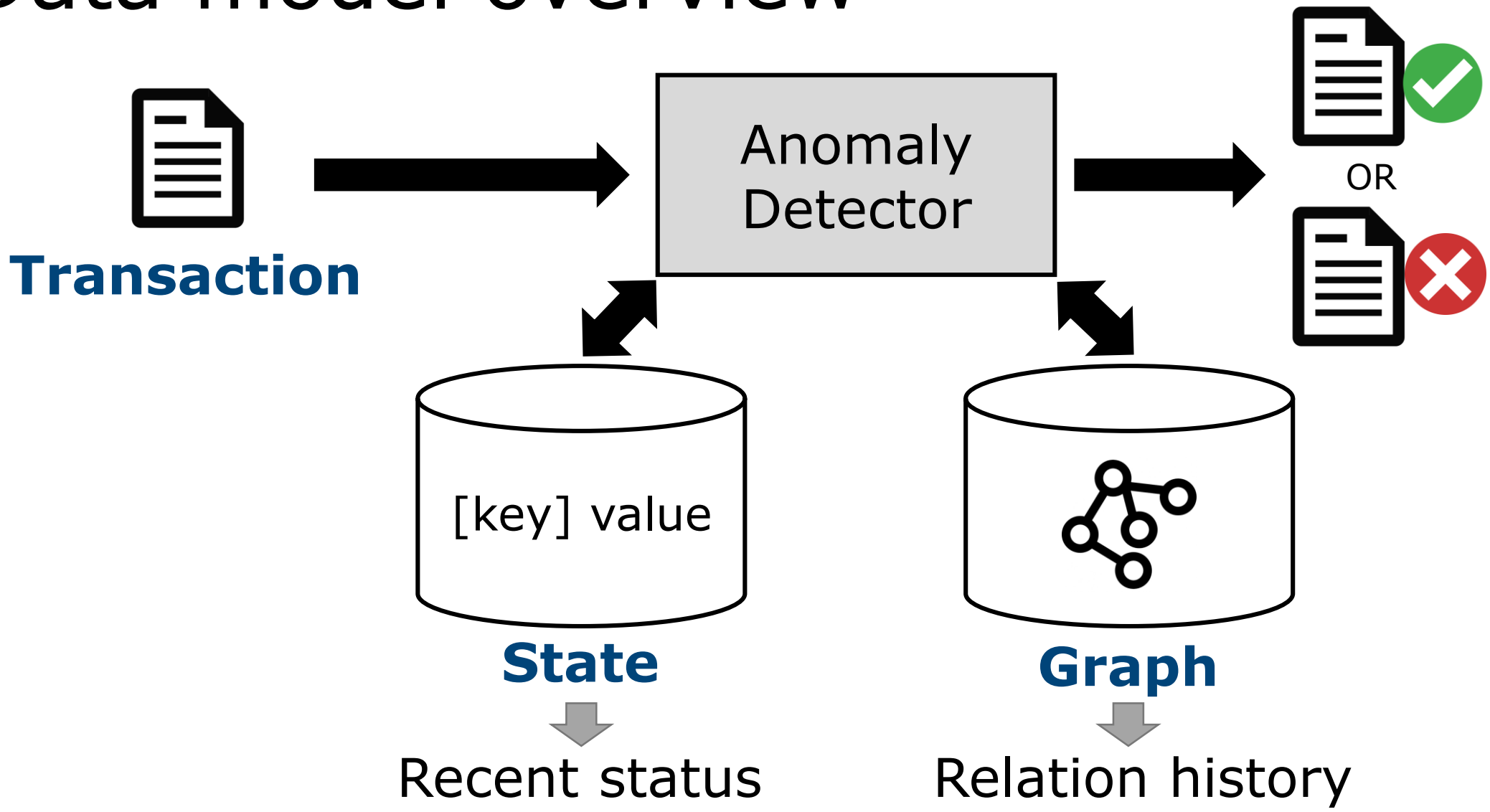
System overview



Outline

- Motivation and goals
- System overview
- Data model
- Verification of business logic
- Implementation and evaluation

Data model overview



Transaction model

- Transaction: supply chain **event**

EPCIS STANDARD
What: Identifiers of associated items
When: Time of an event
Where: Location of an event
Why: business step

An event

Transaction

Tx header	Header signature
Payload	

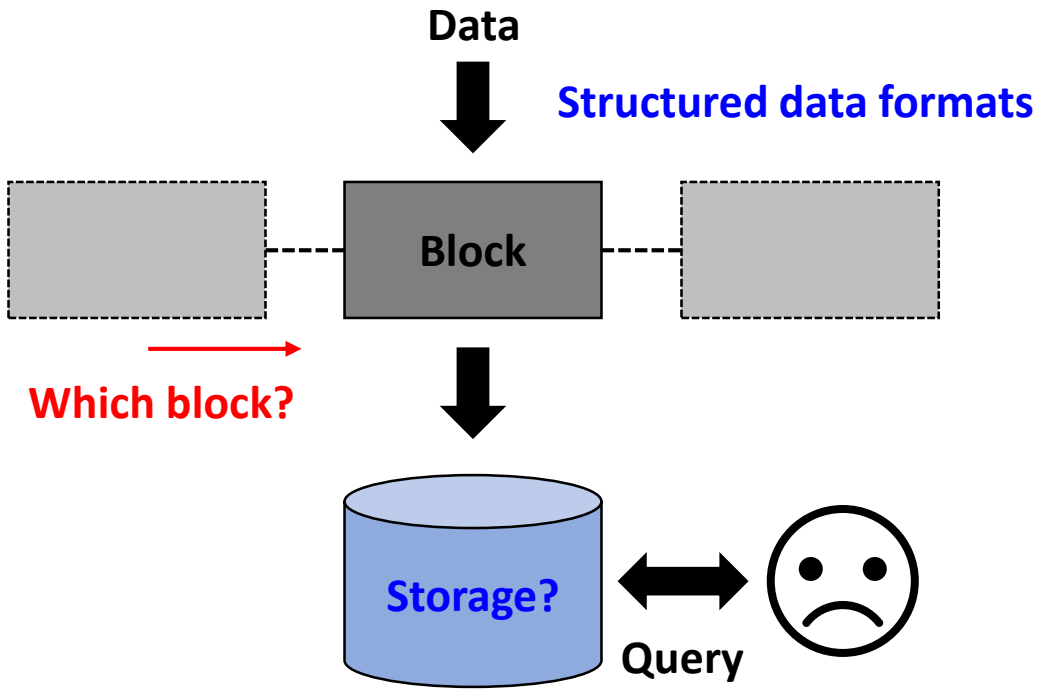
Event_type: aggregation
Product_in: identifier_strawberry (sgtin)
Product_out: identifier_pallet (sscc)
Timestamp: 2019-04-09T13:30:10
Location:identifier_factory (sgln)
Bizstep: loading

State model

- State: current status of an item in supply chain
  - Key-value store built with addressable radix Merkle tree
  - Example
    - Key: identifier of a box of strawberries
    - Value: a tuple (“2019-09-26-10:30”, identifier of a farm, “produced”, “object event”)

Limits of blockchain data model

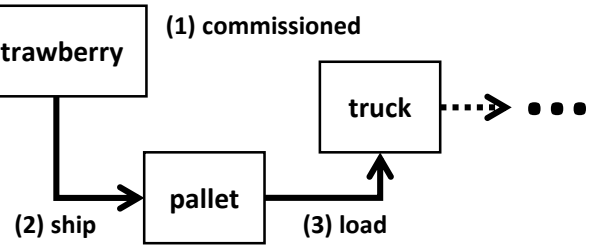
- Current blockchain data model is not adequate for searching data

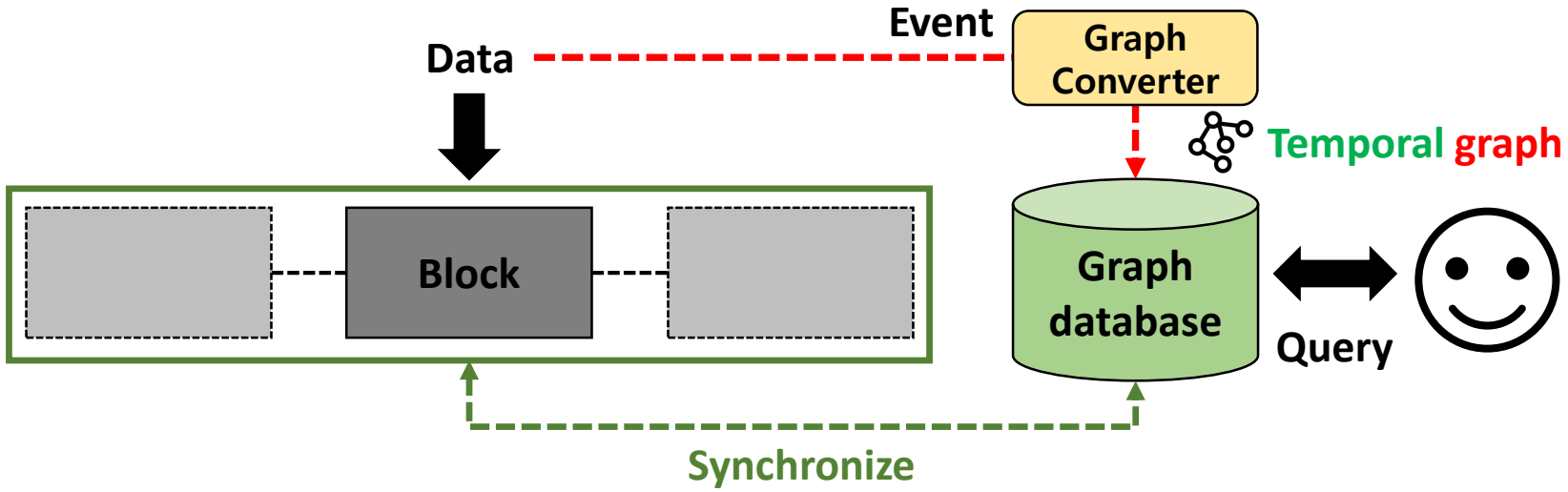


Graph data model

- Main features of supply chain data

Dynamic relationship b/w items along life cycle of a product

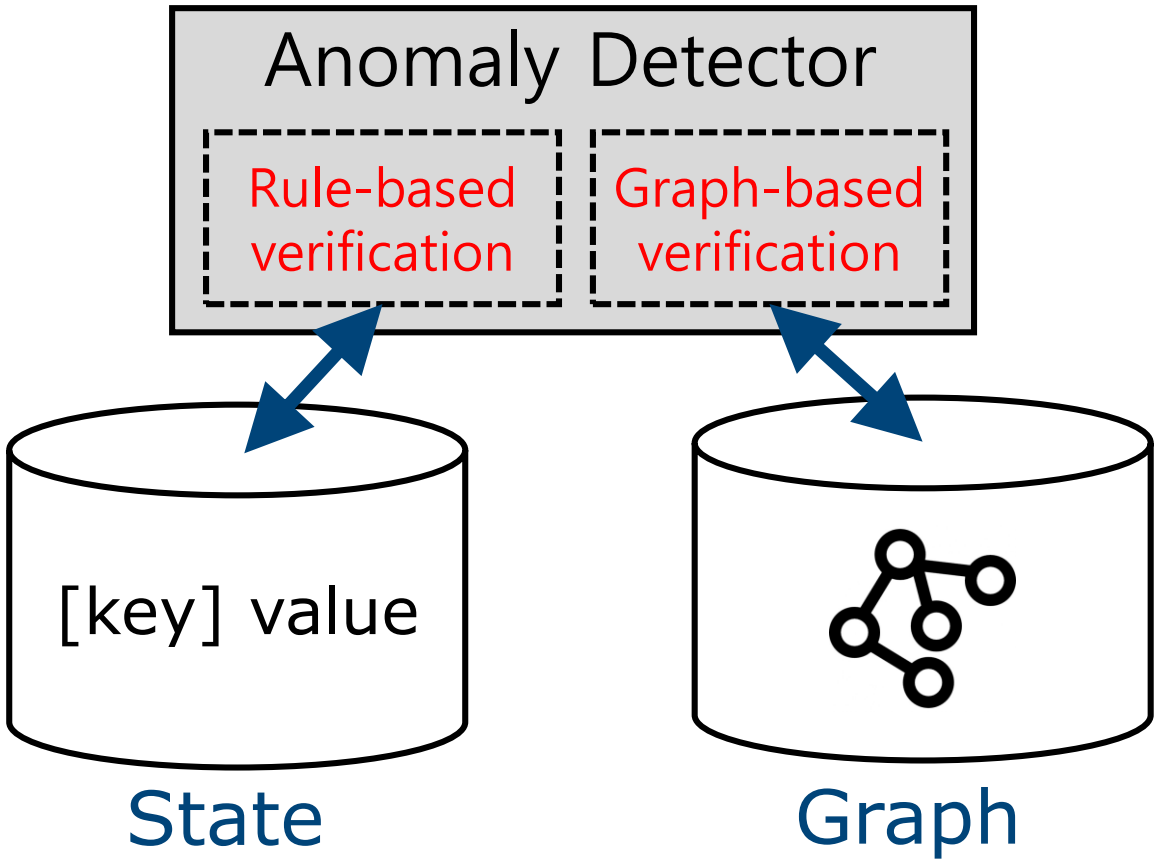




Outline

- Motivation and goals
- System overview
- Data model
- Verification of business logic
- Implementation and evaluation

Verification model overview



Verification models

- Concept of **consistency conditions** in supply chain borrowed from [Ilic et al., 2009]

- Rule-based verification
  - Velocity consistency
  - Dwell-time consistency

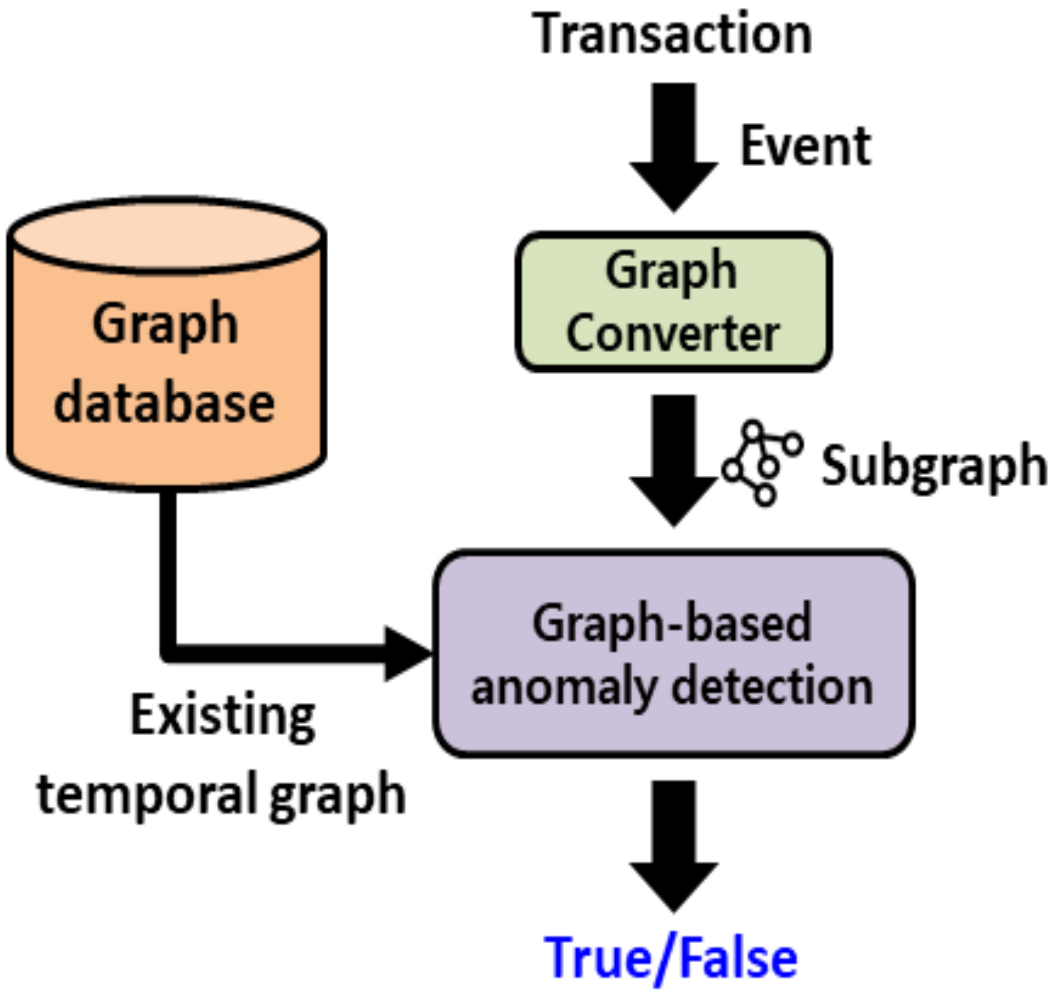
Straightforward solution

1. Configure *threshold* information
2. Inspect state to check if the current data is consistent with the *threshold*

- Graph-based verification
  - Lifecycle consistency

Nontrivial!

Graph-based verification



Lifecycle consistency

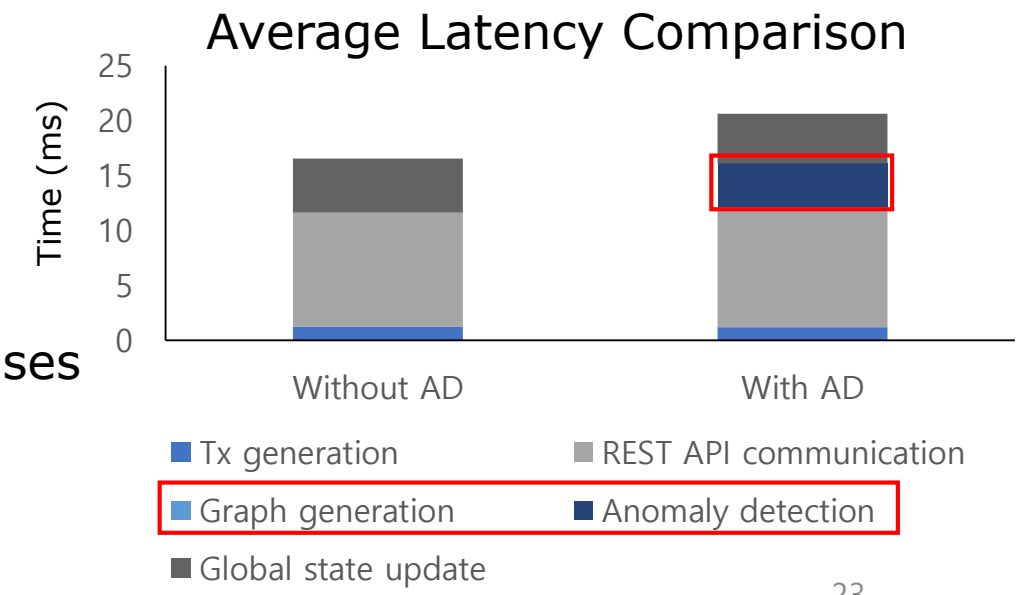
- Check that pairwise business steps are accurately coupled
- Steps:
  - Convert an event to a subgraph
  - Find a matching vertices, traverse through predecessors, and verify the ordering sequences

Outline

- Motivation and goals
- System overview
- Data model
- Verification of business logic
- Implementation and evaluation

Implementation and Evaluation

- Hyperledger Sawtooth
  - Transaction processor that handles **anomaly detection**
- For graph processing, RDFLib is used
  - RDF object generator, serializer (turtle), and persistence database
- Scenario-based simulation
- Correctness
  - Showed deserialized states and transaction processor logs for both normal and abnormal cases
- Performance overheads
  - Compared average latency



Summary

- Blockchain-based SCM is gaining attentions
- Our focus: robust data model + verification methods
- Data model based on real-world business standards tailored for blockchain
- Anomaly detection with rule-based and graph-based approaches