Blockchain-Enabled Certificate-Based Authentication for Vehicle Accident Detection and Notification in IoT-enabled IntelligentTransportation Systems

Dr. Ashok Kumar Das

IEEE Senior Member
Associate Professor
Center for Security, Theory and Algorithmic Research
International Institute of Information Technology, Hyderabad

E-mail: *ashok.das@iiit.ac.in*
URL: `http://www.iiit.ac.in/people/faculty/ashokkdas`
`https://sites.google.com/view/iitkgpakdas/`

Case Study: "Blockchain-Enabled Certificate-Based Authentication for Vehicle Accident Detection and Notification in IntelligentTransportation Systems"

Anusha Vangala, Basudeb Bera, Sourav Saha, Ashok Kumar Das, Neeraj Kumar, and YoungHo Park. "Blockchain-Enabled Certificate-Based Authentication for Vehicle Accident Detection and Notification in Intelligent Transportation Systems," in ***IEEE Sensors Journal***, Vol. 21, No. 14, pp. 15824-15838, July 2021, DOI: 10.1109/JSEN.2020.3009382. (2020 SCI Impact Factor: 3.301)

Introduction

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

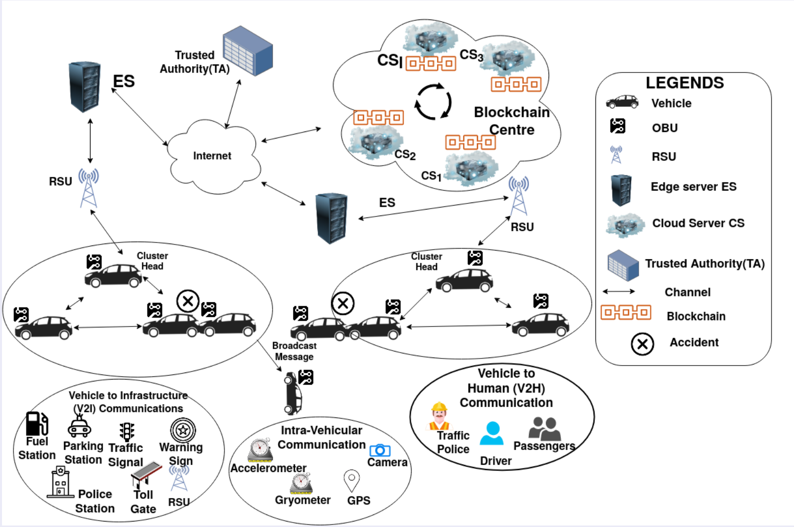

Figure: Blockchain-enabled edge computing based Intelligent Transportation System (ITS)

Motivation

- According to the report cited in WHO 2018, the number of fatalities caused by the road traffic accidents reached 1.35 millions in the year 2016.
- The following statistics show on the number of deaths and the rate of deaths per 10,000 population in traffic accidents world-wide.

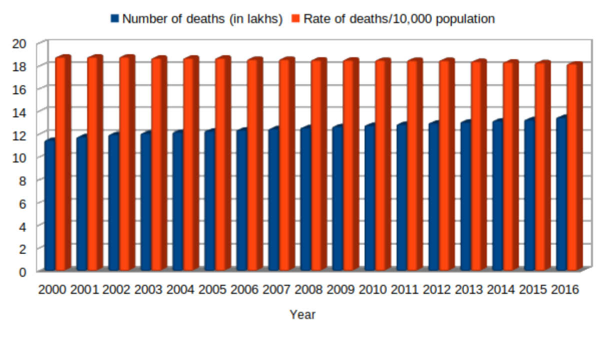

Figure: Number and rate of road traffic deaths during 2000-2016

Motivation (Continued...)

- The communications among various entities in an ITS environment take place via public channels as in other networking environments.
- An adversary can then take opportunity to tamper with the vehicle accident related important data in between the communication among the entities and also mount various potential attacks, such as replay, impersonation, man-in-the-middle, privileged-insider, etc.
- To handle these issues, we aim to design a new blockchain-enabled certificate-based authentication scheme for vehicle accident detection and notification in ITS (BCAS-VADN).

Threat Model

- We contemplate the significantly used "Dolev-Yao threat model (known as the DY model)" in the networking environment.
- Under the DY model, an adversary $\mathcal{A}$ has the ability not only to intercept the communication messages between any two participant (i.e., V_i and CH, CH and RSU_l, and RSU_l to ES_m), but can also inject the malicious messages in the communication channel apart from altering and deleting the contents in the transmitted messages.
- We also consider "Canetti and Krawczyk's model (CK-adversary model)", which is more powerful model as compared to the DY model. In the CK-adversary model, an adversary $\mathcal{A}$ has the capability to compromise the secret credentials, and hijacking the session keys and session states in a particular ongoing session between two parties in the network.
- Furthermore, a vehicle's *OBU* may be physically captured by $\mathcal{A}$. Then, $\mathcal{A}$ can extract all the loaded information from the compromised *OBU*'s memory by utilizing the sophisticated "power analysis attacks"

Notations

Symbol	Meaning
RA	Trusted registration authority
RSU_l, RID_{RSU_l}	l^{th} road-side unit and its pseudo identity
V_i, OBU_i, RID_{V_i}	i^{th} vehicle, its On-Board Unit (OBU) and pseudo identity
CS_w, RID_{CS_w}	w^{th} cloud server and its pseudo identity
ES_m, RID_{ES_m}	m^{th} edge server and its pseudo identity
$n_v, n_{rsu}, n_{es}, n_{cs}$	Number of vehicles, RSUs, edge servers and cloud servers, respectively
$Cert_X$	Certificate of an entity X created by the RA
$h(\cdot)$	A “collision-resistant cryptographic one-way hash function”
$EC(\cdot)/DC(\cdot)$	“Symmetric encryption/decryption functions”, respectively
$EP(\cdot)/DP(\cdot)$	“Public key encryption/decryption functions”, respectively
q	A large prime number
$GF(q)$	Galois finite field over prime q
$E_q(\alpha, \beta)$	A “non-singular elliptic curve: $y^2 = x^3 + \alpha x + \beta \pmod{q}$”
G	A base point G in $E_q(\alpha, \beta)$
$P + Q$	“Elliptic curve point addition” of two points $P, Q \in E_q(\alpha, \beta)$,
$k \cdot G$	“Elliptic curve point multiplication”; $k \cdot G = G + G + \cdots G$ (k times), $G \in E_q(\alpha, \beta)$, $k \in Z_q^*$
$(cpr_X, CPub_X)$	Certificate private-public key pair of an entity X
(pr_X, Pub_X)	Encryption/signature private-public key pair of an entity X
TS_x	Current system timestamp generated by an entity X
ΔT	Maximum transmission delay associated with a message
$x * y$	Modular multiplication of elements $x, y \in Z_q$
$\|\|$	Data concatenation operator
$\oplus$	Bitwise exclusive-OR operator

Proposed Blockchain-Based Authentication Scheme (BCAS-VADN)

The proposed scheme consists of the following modules:

- System Initialization Phase
- Registration/Enrollment Phase
- Authentication Phase
- Blockchain Verification and Addition Phase

User Registration/Enrolment Phase

Vehicle (V_i)
RID_{V_i}, $Cert_{V_i}$, ppr_{V_i}
RSU (RSU_l)
RID_{RSU_l}, $Cert_{RSU_l}$, K_{RSU_l,ES_m}, pr_{RSU_l}
Edge Server (ES_m)
RID_{ES_m}, $Cert_{ES_m}$, pr_{ES_m}, K_{RSU_l,ES_m}
Cloud Server (CS_w)
RID_{CS_w}, $Cert_{CS_w}$, pr_{CS_w}

Figure: Credentials stored in registered entities

Authentication Phase: Summary of authentication between CH and RSU_l

Cluster head (CH)	Road Side Unit (RSU_l)
Generate random secret $r_{CH_2} \in Z_q^*$, current timestamp TS_{CH_2}. Calculate $S_{CH} = h(r_{CH_2}\|\|ppr_{CH}\|\|TS_{CH_2}) \cdot G$, $Sign_{r_{CH_2}} = h(r_{CH_2}\|\|ppr_{CH}\|\|TS_{CH_2}) + h(RID_{CH}\|\|CPub_{CH}\|\|CPub_{RSU_l}\|\|Cert_{CH}\|\|Pub_{RA}\|\|TS_{CH_2}) * ppr_{CH} \pmod{q}$. $Msg_{CHRSU_1} = \{RID_{CH}, Cert_{CH}, S_{CH}, Sign_{r_{CH_2}}, TS_{CH_2}\}$ $\longrightarrow$ (via public channel)	
	Check validity of TS_{CH_2}. Accept/Reject? If so, verify $Cert_{CH}$ as $Cert_{CH} \cdot G \overset{?}{=} Pub_{RA} + h(RID_{CH}\|\|CPub_{CH}\|\|Pub_{RA}) \cdot CPub_{CH}$ If valid, verify signature as $Sign_{r_{CH_2}} \cdot G \overset{?}{=} S_{CH} + h(RID_{CH}\|\|CPub_{CH}\|\|CPub_{RSU_l}\|\|Cert_{CH}\|\|Pub_{RA}\|\|TS_{CH_2}) \cdot PK_{CH}$ If valid, CH is authenticated by RSU_l. Generate random secret $r_{RSU} \in Z_q^*$, current timestamp TS_{RSU}. Compute $T_{RSU} = h(r_{RSU}\|\|pr_{RSU_l}\|\|TS_{RSU}) \cdot G$, $DHK_{RSU,CH} = h(r_{RSU}\|\|pr_{RSU_l}\|\|TS_{RSU}) \cdot S_{CH}$, $\gamma = h(K_{RSU_l,ES_m}\|\|pr_{RSU_l}\|\|Cert_{CH}\|\|Cert_{RSU_l}\|\|TS_{RSU})$, $V_{RSU} = \gamma \oplus h(DHK_{RSU,CH}\|\|Sign_{r_{CH_2}}\|\|TS_{RSU})$, signature on r_{RSU} and $DHK_{RSU,CH}$ as $Sign_{r_{RSU}} = h(r_{RSU}\|\|pr_{RSU_l}\|\|TS_{RSU}) + h(RID_{RSU_l}\|\|Cert_{RSU_l}\|\|DHK_{RSU,CH}\|\|Pub_{RA}\|\|TS_{RSU}) \cdot pr_{RSU_l} \pmod{q}$. $Msg_{CHRSU_2} = \{RID_{RSU_l}, Cert_{RSU_l}, T_{RSU}, V_{RSU}, Sign_{r_{RSU}}, TS_{RSU}\}$ $\longleftarrow$ (via public channel)
Check validity of TS_{RSU}. Accept/Reject? If so, verify if $Cert_{RSU_l} \cdot G \overset{?}{=} Pub_{RA} + h(RID_{RSU_l}\|\|CPub_{RSU_l}\|\|Pub_{RA}) \cdot CPub_{RSU_l}$ If valid, compute $DHK_{CH,RSU} = h(r_{CH_2}\|\|ppr_{CH}\|\|TS_{CH_2}) \cdot T_{RSU}$, $\gamma' = h(K_{RSU_l,ES_m}\|\|pr_{RSU_l}\|\|Cert_{CH}\|\|Cert_{RSU_l}\|\|TS_{RSU}) = V_{RSU} \oplus h(DHK_{CH,RSU}\|\|Sign_{r_{CH_2}}\|\|TS_{RSU})$. If valid, verify signature $Sign_{r_{RSU}}$ as $Sign_{r_{RSU}} \cdot G \overset{?}{=} T_{RSU} + h(RID_{RSU_l}\|\|Cert_{RSU_l}\|\|DHK_{CH,RSU}\|\|Pub_{RA}\|\|TS_{RSU}) \cdot CPub_{RSU_l}$. If signature is valid, RSU_l is authenticated by CH. Generate current timestamp TS_{CH_3} and compute $SK_{CH,RSU} = h(DHK_{CH,RSU}\|\|\gamma'\|\|Sign_{r_{CH_2}}\|\|Sign_{r_{RSU}}\|\|TS_{CH_3})$, session key verifier $SKV_{CH,RSU} = h(SK_{CH,RSU}\|\|TS_{CH_3})$. $Msg_{CHRSU_3} = \{SKV_{CH,RSU}, TS_{CH_3}\}$ $\longrightarrow$ (via public channel)	
	Check validity of TS_{CH_3}. Accept/Reject? If valid, compute $SK_{RSU,CH} = h(DHK_{RSU,CH}\|\|\gamma\|\|Sign_{r_{CH_2}}\|\|Sign_{r_{RSU}}\|\|TS_{CH_3})$. Check $SKV_{RSU,CH} \overset{?}{=} h(SK_{RSU,CH}\|\|TS_{CH_3})$. If so, session key is considered as valid.
Both CH and RSU_l store the shared session key $SKV_{CH,RSU} (= SKV_{RSU,CH})$	

Block formation

- The blocks are created in two levels: 1) partially by an edge server (ES_m) and 2) full block by a cloud server (CS_w) in the BC.
- A vehicle (V_i) first creates a transaction, say Tx_i, once a vehicle accident (either the same vehicle or its nearby neighbor vehicle(s)) is detected by its own OBU_i.
- The format of Tx_i contains the following fields: a) time of accident takes place ($Time_{acd}$), b) location of accident (Loc_{acd}), c) ID of the vehicle in accident (ID_{Vacd}), d) ID of reporting vehicle (ID_{Vrep}), e) direction of accident vehicle (Dir_{Vacd}), f) position of accident vehicle (Pos_{Vacd}), g) level of accident vehicle ($Level_{Vacd}$) and h) severity of the passengers in the accident vehicle (Sev_{Vacd}), which can be either "no injury" or "non-incapacitating injury" or "incapacitating or fatal injury".
- Based on the direction and position, the accident is further classified into three levels: a) minor, b) moderate and c) severe

Block formation

- Next, V_i will generate a signature Sig_{Tx_i} using the signature generation algorithm of the "Elliptic Curve Digital Signature Algorithm (ECDSA)" on the message $msg = h(Tx_i)$ with the help of its own private key ppr_{V_i}. After that V_i will use the already established session key $SK_{V_i,CH}$ with its neighbor CH to send the notification message $Msg_{notif} = \langle RID_{V_i}, EC_{SK_{V_i,CH}}(Tx_i, RID_{V_i}), Sig_{Tx_i}\rangle$ to CH via public channel.
- After receiving Msg_{notif}, CH decrypts $EC_{SK_{V_i,CH}}(Tx_i, RID_{V_i})$ using the established session key $SK_{V_i,CH}$ shared with V_i to retrieve $(Tx_i, RID'_{V_i}) = DC_{SK_{V_i,CH}}[EC_{SK_{V_i,CH}}(Tx_i, RID_{V_i})]$ and check if $RID'_{V_i} = RID_{V_i}$. If it is valid, CH validates the signature Sig_{Tx_i} on $msg = h(Tx_i)$ using the ECDSA signature verification algorithm. If the signature is valid, CH sends the information $\{Tx_i, Sig_{Tx_i}\}$ securely using the secret key $SK_{CH,RSU}$ to its associated RSU_l.
- RSU_l validates the Sig_{Tx_i} on received transaction Tx_i using ECDSA signature verification algorithm. If the signature is valid, RSU_l sends securely the information $\{Tx_i, Sig_{Tx_i}\}$ to its associated ES_m using the pre-shared key K_{RSU_l,ES_m}.

Partial Block Formation

- If the signature Sig_{Tx_i} is validated successfully on the received Tx_i, ES_m stores (Tx_i, Sig_{Tx_i}). After filtering all the transactions, assume that ES_m has a list of n_t important transactions which need to be applied in block formation.
- ES_m then creates a partial block, say $PartialBlock_i$, containing n_t transactions Tx_i and its signature Sig_{Tx_i}, which has the formats shown in Fig.4. The Merkle tree root (MTR_i) is obtained from the n_t transactions Tx_i ($i = 1, 2, \cdots, n_t$). Next, $PartialBlock_i$ is forwarded to its respective cloud server CS_w.

Block Header	
Merkle Tree Root	MTR_i
Owner of Block	OB_i (ES_m)
Public key of signer	Pub_{ES_m}
Block Payload (Transactions)	
List of n_t transactions and their signatures	$\{(Tx_i, Sig_{Tx_i}) \mid i = 1, 2, \cdots, n_t\}$
Signature on all transactions (Tx_i)	Sig_{Block_i}

Figure: Structure of a partial block $PartialBlock_i$ created by edge server ES_m

Full Block formation

- After receiving $PartialBlock_i$ from ES_m, CS_w will make the full block $Block_i$ on $PartialBlock_i$, by adding the timestamp, unique block version, previous hash block (PBH_i) and current hash block (CBH_i), where $CBH_i = h(\text{Block Header} \,||\text{Block Payload})$.

Block Header	
Block Version	$BVer_i$
Previous Block Hash	PBH_i
Timestamp	TS_i
Merkle Tree Root	MTR_i
Owner of Block	$OB_i\ (ES_m)$
Public key of signer	Pub_{ES_m}
Block Payload (Transactions)	
List of n_t transactions and their signatures	$\{(Tx_i, Sig_{Tx_i}) \,\vert i = 1, 2, \cdots, n_t\}$
Signature on all transactions (Tx_i)	Sig_{Block_i}
Current Block Hash	CBH_i

Figure: Structure of a full block $Block_i$ created by a cloud server CS_w

Consensus for Block Verification and Addition

Algorithm 1 Consensus algorithm for a block ($Block_i$) verification and addition in blockchain

Input: $Block_i$ = {Block Header, Block Payload, CBH_i}; private-public key pairs (pr_{CS_w}, $Pub_{CS_w} = pr_{CS_w} \cdot G$) for all cloud servers CS_w; f_{cs}.

Output: Addition of $Block_i$ in the blockchain after successful validation.

1: Let CS_w select a leader, say CS_L using the existing leader selection algorithm [42].
2: CS_L creates a voting request, say $VReq$.
3: CS_L generates a random nonce r_{CS} and creates a signature Sig_{CS_L} using its own private key pr_{CS_L} on $h(Block_i \,||r_{CS}\,||VReq\,||Cert_{CS_L})$.
4: CS_L sends the request message $\langle Block_i, Sig_{CS_L}, EP_{Pub_{CS_P}}[r_{CS}, VReq], Cert_{CS_L}\rangle$ to other peer cloud servers CS_P in the P2P CS network via public channel.
5: Each peer CS_P after getting request message, computes $(r_{CS}, VReq) = DP_{pr_{CS_P}}[EP_{Pub_{CS_P}}[r_{CS}, VReq]]$, verifies $Cert_{CS_L}$ and Sig_{CS_L} on $Block_i$, r_{CS}, $VReq$ and $Cert_{CS_L}$ using Pub_{CS_L}.
6: If both certificate and signature are valid, CS_P further verifies MTR_i, CBH_i, and Sig_{Block_i} on the received $Block_i$.
7: If all these validations are successful, CS_P prepares the voting response, $VRes$ and generates the ECDSA signature Sig_{CS_P} created on $h(r_{CS}\,||VRes\,||VReq\,||Cert_{CS_P})$ using pr_{CS_P}.
8: CS_P sends the response message $\langle EP_{Pub_{CS_L}}[VRes], Sig_{CS_P}, Cert_{CS_P}\rangle$ to CS_L via public channel.
9: Let $ValidVC$ denote the valid vote counter. Set $ValidVC \leftarrow 0$.
10: **for** each received message $\langle EP_{Pub_{CS_L}}[VRes], Sig_{CS_P}, Cert_{CS_P}\rangle$ from CS_L's followers CS_P **do**
11: CS_L validates $Cert_{CS_P}$ and Sig_{CS_P} using the CS_P's Pub_{CS_P}.
12: CS_L computes $VRes = DP_{pr_{CS_L}}[EP_{Pub_{CS_L}}[VRes]]$.
13: **if** $((Sig_{CS_P} = valid)$ and $(VReq = valid)$ and $(VRes = valid))$ **then**
14: Set $ValidVC = ValidVC + 1$.
15: **end if**
16: **end for**
17: **if** $(ValidVC > 2f_{cs} + 1)$ **then**
18: Send commit response (i.e., $Block_i$ is successfully verified) to all followers CS_P.
19: Add $Block_i$ to the blockchain.
20: **end if**

Security Analysis

The proposed scheme is resilient against various known attacks:

- Impersonation Attacks
- Replay Attack
- Impersonation Attacks
- Man-in-the-Middle Attack
- Privileged-insider Attack
- Physical Vehicle Capture Attack
- Ephemeral Secret Leakage (ESL) Attack

Formal Security Security Verification using Automated Validation of Internet Security Protocols and Applications (AVISPA) tool

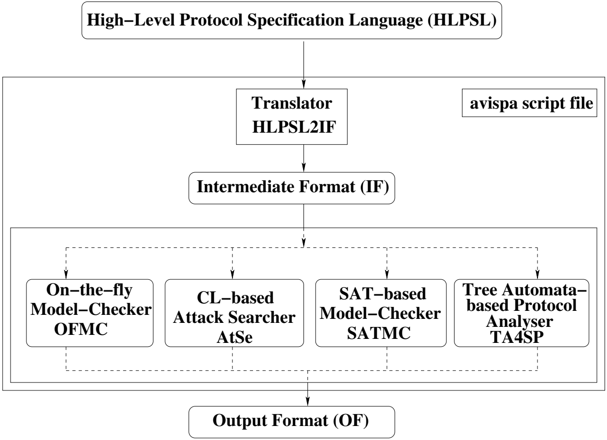

Figure: Architecture of AVISPA (http://www.avispa-project.org/)

Simulation results under OFMC and CL-AtSe backends

Case 1. In this case, we implemented the basic roles for a vehicle (V_i), its associated *CH* and the *RA* during the enrollment and authentication phases.

```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS

PROTOCOL
  /home/anusha/Desktop/span/
  testsuite/results/Case1−V2CH.if

GOAL  as specified

BACKEND  OFMC

STATISTICS
  TIME 10026 ms
  parseTime 0 ms
  visitedNodes: 256 nodes
  depth: 8 plies
```

```
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL
PROTOCOL
  /home/anusha/Desktop/span
  /testsuite/results/Case1−V2CH.if
GOAL
  As specified

BACKEND
  CL−AtSe

STATISTICS
  Analysed   : 39943 states
  Reachable  : 3 states
  Translation: 1.68 seconds
  Computation: 0.11 seconds
```

Simulation results under OFMC and CL-AtSe backends

Case 2. Under this case, we implemented the basic roles for a cluster head (*CH*), its respective RSU_l and the *RA* during the enrollment and authentication phases.

```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS

PROTOCOL
  /home/anusha/Desktop/span/
  testsuite/results/Case2−CH2RSU.if

GOAL  as specified

BACKEND  OFMC

STATISTICS
  TIME 10743 ms
  parseTime 0 ms
  visitedNodes: 256 nodes
  depth: 8 plies
```

```
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL
PROTOCOL
  /home/anusha/Desktop/span/
  testsuite/results/Case2−CH2RSU.if
GOAL
  As specified

BACKEND
  CL−AtSe

STATISTICS
  Analysed   : 39943 states
  Reachable  : 3 states
  Translation: 2.84 seconds
  Computation: 0.12 seconds
```

Blockchain Implementation

- **Scenario 1:** We assume that the total number of peer-to-peer (P2P) nodes in the CS network is 5. It is worth noticing that the computational time (in seconds) differs for the varied number of blocks mined into a blockchain, where each block contains a fixed number of transactions as 60.

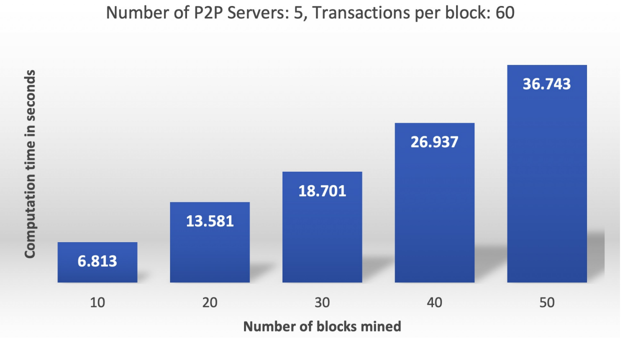

Figure: Blockchain simulation results for Scenario 1

Blockchain Implementation

- **Scenario 2:** In this situation, the total number of peer-to-peer (P2P) nodes in CS network is also considered as 5. We have considered a fixed number of mined blocks as 30. The simulation results show the computational time (in seconds) also differs based on the number of varied transactions pushed per block.

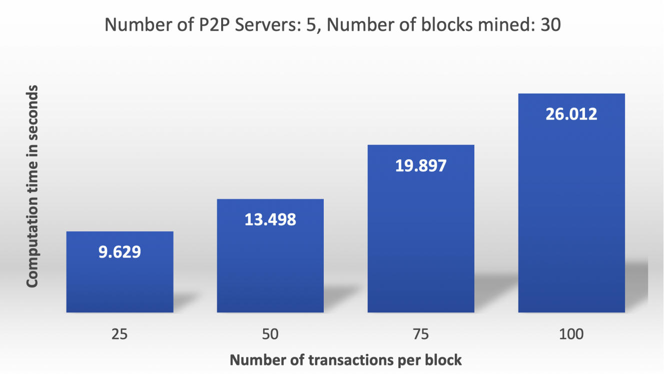

Figure: Blockchain simulation results for Scenario 2

Thank You
For Your Attention