Blockchain-Envisioned Trusted Random Oracles for IoT-Enabled Probabilistic Smart Contracts

Nikunjkumar Sureshbhai Patel, Pronaya Bhattacharya, Shivani Bharatbhai Patel, Sudeep Tanwar, *Senior Member, IEEE*, Neeraj Kumar, *Senior Member, IEEE*, and Houbing Song, *Senior Member, IEEE*

***Abstract*—In modern decentralized Internet-of-Things (IoT)-based sensor communications, pseudonoise-diffusion oracles are heavily investigated as random oracles for data exchange among peer nodes. As these oracles are generated through algorithmic processes, they pass the standard random tests for finite and bounded intervals only. This ensures a false sense of privacy and confidentiality in exchange through open protocol IoT-stacks in public channels, i.e., Internet. Recently, blockchain (BC)-envisioned random sequences as input oracles are proposed about financial applications, and windfall games like roulette, poker, and lottery. These random inputs exhibit fairness, and nondeterminism in SC executions termed as probabilistic smart contracts (PSCs). However, the IoT-enabled PSC process might be controlled and forged through humans, machines, and bot-nodes through physical and computational methods. Moreover, dishonest entities like contract owners, players, and miners can co-ordinate together to form collusion attacks during consensus to propagate false updates, which ensures forged block additions by miners in BC. Motivated by these facts, in this article, we propose a BC-envisioned IoT-enabled PSC scheme, *SaNkhyA*, which is executed in three phases. In the first phase, the scheme eliminates colluding dishonest miners through the proposed miner selection algorithm. Then, in the second phase, the elected miners agree through the proposed consensus protocol to generate a stream of random bits. In the third phase, the generated random bit-stream is split through random splitters and fed as input oracles to the proposed PSC among participating entities. In simulation, the scheme ensures a trust probability of 0.38 even at 85% collusion among miners and has an average block processing delay of 1.3 s compared to serial approaches, where the block processing delay is 5.6 s, thereby exhibiting improved scalability. The overall computation and communication cost is 28.48 ms, and 101 bytes, respectively, that indicates the efficacy of the proposed scheme compared to the traditional schemes.**

***Index Terms*—Blockchain (BC), consensus, Internet-of-Things (IoT), probabilistic smart contracts (PSCs), random numbers.**

Manuscript received November 1, 2020; revised March 23, 2021; accepted April 2, 2021. Date of publication April 9, 2021; date of current version September 23, 2021. *(Corresponding authors: Houbing Song; Sudeep Tanwar.)*

Nikunjkumar Sureshbhai Patel, Pronaya Bhattacharya, Shivani Bharatbhai Patel, and Sudeep Tanwar are with the Department of Computer Science and Engineering, Institute of Technology, Nirma University, Ahmedabad 382481, India (e-mail: 17bce083@nirmauni.ac.in; pronoya.bhattacharya@nirmauni.ac.in; 17bce117@nirmauni.ac.in; sudeep.tanwar@nirmauni.ac.in).

Neeraj Kumar is with the Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Deemed to be University, Patiala 147004, India, also with the Department of Computer Science, King Abdulaziz University, Jeddah 21589, Saudi Arabia, also with the Department of Computer Science and Information Engineering, Asia University, Taichung City 402701, Taiwan, and also with School of Computer Science, University of Petroleum and Energy Studies, Dehradun 248007, India (e-mail: neeraj.kumar@thapar.edu).

Houbing Song is with the Security and Optimization for Networked Globe Laboratory, Department of Electrical Engineering and Computer Science, Embry-Riddle Aeronautical University, Daytona Beach, FL 32114 USA (e-mail: h.song@ieee.org).

Digital Object Identifier 10.1109/JIOT.2021.3072293

I. Introduction

DECENTRALIZED Internet-of-Things (IoT)-based sensor communication among peer nodes involves the generation of random inputs through the pseudocomputational process, exchanged via public wireless channels. These processes are intuitively random noise-based algorithms and follow a predictable pattern of discovery. This induces a false sense of privacy of exchanged sensor data among peer nodes, which can be compromised through intelligent attack vectors [1]. Moreover, the exchanged sensor data needs to be modulated with high-performance networks, so latency and error rates can be decreased [2]. Thus, to address the mentioned limitations, blockchain (BC) can be a potential solution to IoT-enabled random oracles due to its inherent nature of ensuring trust, transparency, immutability, and chronology in mined transactions.

Initially started as cryptocurrency derivatives, BC can leverage the generation of secure random oracles that ensures trust among IoT-based stakeholders. Also, to allow fair, secure, and trusted resource trading among decentralized IoT-ecosystems, BC-envisioned random oracles ensure fair and transparent prices among different players—buyers and sellers in IoT ecosystems. Random sequences are also applicable in online windfall games like lottery, casinos, and pokers, which gained prominence due to government regulations. Windfall games select a game winner from a set of game players (GPs), which is decided by game owners (GOs) in the ecosystem. Once a winner is selected, SC automates anonymous transactional payments among winners and GO based on an agreed set of fed oracles and deterministic conditions as specified in the SC by GP and GO. Thus, in such multiparty lottery games, and IoT-based resource trading scenarios, fairness in the execution of SC, and selection of winners from GP is a critical issue. GO can form biased oracles as fed inputs to SC, and can collude with miners entities to propagate false block additions.

Thus, GO can control game inputs and game environments to ensure SC execute in their favor, that selects a biased winner from the set of GP. Also, deterministic SC are exposed

TABLE I
CHARACTERIZATION OF RANDOM NUMBER GENERATION

Random Number Generation Sources	1	2	3	4
Human	✗	✗	✗	✓
Computational Processes	✓	✗	✓	✓
Bot-Servers	✓	✗	✗	✓
Physical Processes	✓	✓	✓	✗

1.Privacy; 2. Confidentiality; 3. Integrity; 4. Latency

to security flaws like call stack depth limitations [3], underflow/overflow issues [4], and greedy contracts [5]. Due to this, even in secured BC ecosystems, GP feels cheated due to financial losses, nontransparency in-game evaluation, and winner selection. Thus, secure randomness in the generation of inputs to deterministic SC is required, termed as probabilistic smart contract (PSC). PSC ensures oracles achieve nondeterminism, unpredictability, fairness, and verifiability, with a defined probability, on different runs of the same SC.

In PSC, the pseudorandom sequences are created by users through different seed generation methods, such as usage of a hash of current mined block [15], random sequences digitized through physical processes, usage of library functions [16], providing an incentive to users in BC to generate random bits [17]. A security characterization of the random number generation sources is presented in Table I. However, PSC fails to produce fairness and transparency in the selection of oracles as they are nonresistant to collusion attacks by dishonest parties in BC. Collusion attacks can proceed through different ways, such as selection of a fraudulent leader through consensus, confiscating random seed, and fraudulent miner selection through bribe of incentives like computational power, storage, and cryptocurrencies. Thus, necessary countermeasures are required in BC to prevent collusion attacks by dishonest entities. This ensures a fair selection scheme for random oracle to PSC that guarantees trust and in-game correctness outputs. To mitigate the effects of game manipulation by GO, the source of randomness needs to be secured, and consensus mechanism needs to correctly decide game operations and incentives for miners [18]. SC execution conditions must favor random function calls so that dishonest miners are not favored by more block additions. Also, PSC in IoT allows resilient and secure communication stack, with effective coding schemes fitted to sensor microprocessor units [19]. It allows effective true random number seeds, with low-powered hardware encryption costs, and are resilient to hardware-injection, clock-based, and time-delay attacks.

II. STATE OF THE ART

In literature, the author's proposed solutions to randomize SC and to ensure transparency, immutability, privacy, and trust among participating stakeholders in BC. For example, Chatterjee *et al.* [6] proposed a game-theoretic approach to generate immutable pseudorandom sequences in BC. The application environment allows PSC executions through random environment variables that do not rely on consensus on selected miners. Du *et al.* [7] proposed node participation for random number generation to improve trust and performed experiments that validated security and privacy computations. However, inducing trust as a parameter in SC execution was not addressed. Nguyen Van *et al.* [8] proposed generation of verifiable random numbers based on homomorphic encryption that generates unpredictable, and immutable random numbers with public access. Mulders [20] proposed a scheme to generate randomization environment on Ethereum based on parameters like *eth.blockstamp*, and *eth.timestamp*. However, if the generated random number is not favorable in malicious miner, the number is not submitted for consensus.

Choi *et al.* [9] proposed a system to generate random seed using out-of-band communication and hardware variation for the use of IoT. Li *et al.* [10] proposed a random lottery-based smart contract scheme that ensures winning sequences for GPs to be generated fairly. Wang *et al.* [11] reviewed the bitcoin transaction having vulnerability because of weak randomness.To overcome the same, Datta [12] proposed a secure pseudorandom generation scheme on point-based GF(p) encryption for collusion attacks. However, the time complexity of determining a set of *GF(p)* over arbitrary polynomial curves is exponential, rendering the approach nonscalable. Ehara and Tada [13] proposed a transparent random number generation scheme on public BC. To frame out the consensus among stakeholders, the authors used the proof-of-work consensus scheme, which is resource-intensive for constrained environments. This issue was addressed by Feng *et al.* [14] in which they proposed a new consensus protocol proof of negotiation to randomize the selection of miners, which eliminates the collusion among dishonest miners and achieves high throughput in block creation. A summary of the state-of-the-art works based on the proposed techniques and simulation environments to generate random sequences is shown in Table II.

A. Motivation

As discussed in Section II, ensure fairness and transparency in-game operations among GP and GO, secure randomization in input oracles to SC [6], [7] is necessary. Moreover, collusion among dishonest miners forces nontransparency in SC evaluation, which ensures forged block additions in favor of dishonest entity. Incorrect oral updates are propagated in BC, which results in compromised consensus among participating entities. Thus, the authors in [12] and [13] proposed a stable election scheme that eliminates colluding miners and allows honest block additions. To design a random scheme to secure oracles, and eliminate dishonest entities, the proposed scheme, *SaNkhyA*, exploits a model that ensures triple benefits-elimination of dishonest miners, validation of honest miner selection through a stable consensus protocol termed as Proof-of-Validity (PoV), and selection of random bit-sequences as outputs of PoV to fed to SC. Thus, the scheme achieves nondeterminism in fed oracles to PSC and eliminates the risk of bribery attacks by colluding miners in parallel. This allows an end-to-end solution of transparent operations through correct block updates in the proposed ecosystem.

B. Research Contributions

Following are the research contribution of this article.

1) A BC-envisioned scheme is proposed to ensure the immutability and transparency of events among entities

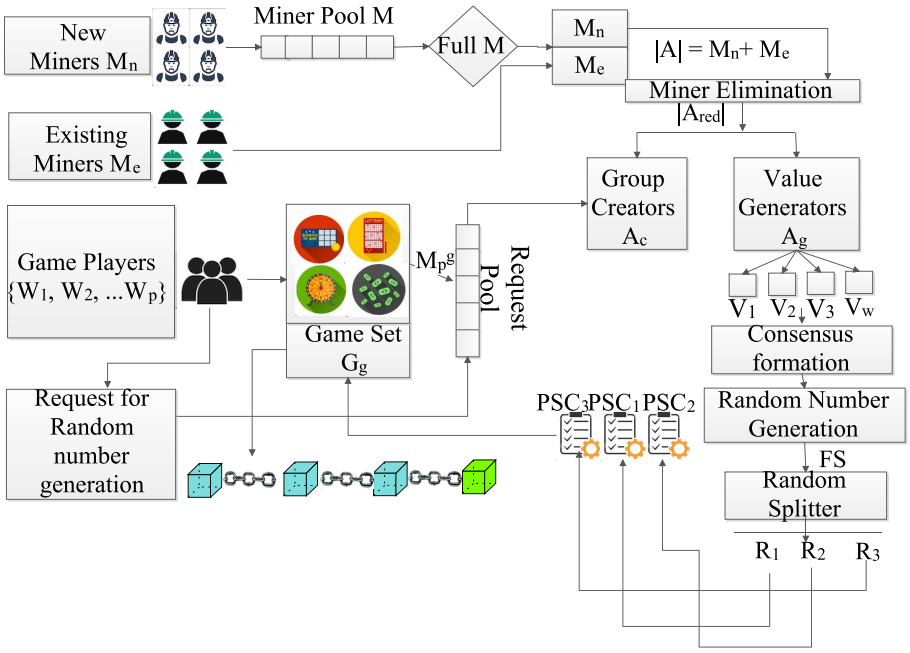


Fig. 1. *SaNkhyA*: the system model.

TABLE II
RELATIVE COMPARISON OF STATE-OF-THE-ART EXISTING LITERATURE

Authors	Miner elimination	Consensus	Collusion safe	Random oracles	PSC	Technique used	Environment
Chatterjee *et al.* [6]	✗	✓	✓	✓	✓	Incentive game-theory to generate random numbers	Smart Contracts
Du *et al.* [7]	✓	✓	✓	✓	✗	BC-based random game theory to prevent human manipulation	Windfall games
Nguyen-van *et al.* [8]	✗	✓	✗	✓	✓	Verifiable and scalable random number generation	Financial Contracts
Choi *et al.* [9]	✗	✗	✗	✓	✗	Random seed for IoT-based device encryption scheme	Industrial-IoT
Li *et al.* [10]	✓	✗	✗	✓	✓	PSC based on winning-random sequence generation	Windfall games
Wang *et al.* [11]	✗	✓	✗	✓	✗	Securing bitcoin block additions through random seed	Cryptocurrency schemes
Datta *et al.* [12]	✓	✓	✗	✓	✗	Constrained pseudo-random functions over GF(p)	IoT-based encryption schemes
Ehara *et al.* [13]	✗	✓	✓	✓	✗	Transparent random number based on PoW	Cryptocurrency schemes
Feng *et al.* [14]	✓	✓	✓	✓	✗	Consensus scheme PoN	Windfall games
SaNkhyA	✓	✓	✓	✓	✓	Elimination of dishonest miners with proposed consensus Proof-of-Validity (PoV)	Financial applications, and windfall games.

GP and GO, and facilitates effective random oracle generation among resource players in IoT-enabled trading ecosystems.

2) A random miner selection algorithm is proposed to eliminate dishonest miners and PoV consensus protocol to agree on selected random number $\mathbb{R}$ as a sequence of bits.
3) Based on stream of bits $\mathbb{R}$, a random splitter mechanism is used to split the bit-stream as subsequences to PSC, which is to be executed among GP and GO to ensure fairness and transparency in-game events.

C. Layout

This article is presented as follows. Section III presents the system model and the problem formulation. Section IV discusses the proposed scheme for the generation of random number $\mathbb{R}$ and proposed PSC. Section V discusses the performance evaluation and finally, Section VI concludes this article.

III. *SaNkhyA*: SYSTEM MODEL AND PROBLEM FORMULATION

This section describes the system model and the problem formulation.

A. System Model

A BC-based scheme, *SaNkhyA*, is proposed to ensure fair oracles to PSC and to leverage the trust among GP and GO as shown in Fig. 1. The generation of random oracles

are proposed to allow fair, trusted and secure trading among resource traders in IoT ecosystems. Also, in windfall games, random numbers are an important choice for in-transparency in game events. In this scheme, E_{GP} and E_{GO} represents the GP and GO, respectively. Moreover, E_{GP} is denoted by the set $\{W_1, W_2, \ldots, W_p\}$ and E_{GO} by $\{O_1, O_2, \ldots, O_g\}$. The game-set is denoted as a set $G = \{G_1, G_2, \ldots, G_g\}$ owned by GO.In any g^{th} game, E_{GP} generates a registration request R to E_{GO}, that on request acceptance, forms a mapping $M : W_p \leftarrow G_g$, that denotes W_p information to be stored with game-set G. G contains metadata of R to be stored in BC B denoted as $M_p = \{W_p^g, T_p^g\}$, i.e., wallet and timestamp information. After the successful registration, W_p makes a request for generation of a random number $\mathbb{R}$ for the game G_g based on M_p as an input from B. To ensure fairness in generation of $\mathbb{R}$, we consider entity E_M as miners, which consists of new miners M_n and existing miners M_e in the scheme. E_M participates in generation of $\mathbb{R}$ as fed random oracles. To address the same, M_n are added to a miner pool M and once this pool is full then both M_e and M_n can participate in the proposed miner elimination algorithm to allow only honest miners in the scheme. This algorithm takes trust factor TR for each miner as inputs based on their past participation and activities in B. If TR is less than the minimum trust threshold TR_{min} for any E_M, then the corresponding miner is eliminated in process of generation of random $\mathbb{R}$.

Based on miner elimination process as explained above, a reduced set A_{red} is formed. From this set, we can create two entities termed as- group creators A_c and value generators A_g. A_c forms a request pool of W_p as M_p^g. M_p^g requests $\mathbb{R}$ to be generated for any g^{th} game from game-set G. A_c assigns metadata M_p collected from M_p^g to A_g in a first-come-first serve (FCFS) manner. A_g assign a value V to each request. Collectively all A_g execute a consensus procedure base on the value V generated for requests and agree on a random number $\mathbb{R}$, which is outputted as a continuous stream of bit-sequence $\{b_o, b_1, b_2, \ldots, b_n\}$ denoted as final string FS. Then, this final string is passed through a random splitter, which divides $\{b_0, b_1, b_2, \ldots, b_n\}$ into equal-sized substring of bits represented as $\{R_1, R_2, \ldots, R_n\}$ where each R_i is a subsequence $\{b_k, b_{k+1}, \ldots, b_l\}$, with conditions $0 \leq k \leq (n-1)$, $1 \leq l \leq n$ and $k < l$, respectively. The generated FS serves as an input oracles to PSC set $\{\text{PSC}_1, \text{PSC}_2, \ldots, \text{PSC}_k\}$. Thus, random splitter allows k subsequences for $\forall \text{PSC}_k$. As inputs are random, PSC_k execution varies for different runs on same R_n, exhibiting nondeterminism. Also, as $\mathbb{R}$ is generated through consensus, the input R_n achieves trust among E_{GP} and E_{GO}. Post execution of PSC, the wallet balances of E_{GP} and E_{GO} are updated to indicate that winning amount is transferred to the respective winner W_g as per their bet for game G_g.

B. Problem Formulation

In *SaNkhyA* as explained in Section III-A, to formulate the problem, we consider p players are mapped to g games through mapping M. The metadata M_p consists of wallet W_p^g information with the following attributes:

$$W_p^g = \left\{\text{PK}_p^g, \text{ID}_p^g, \text{TX}_p^g, \text{mroot}\right\} \tag{1}$$

where PK_p^g is public key of the p^{th} GP in g^{th} game, ID_p^g is the identifier information of p^{th} GP for authentication, TX_p^g denotes the set of transactional entries among different players p_i and p_j involved in G_g, with $1 \leq i, j \leq p$ and $i \neq q$, and mroot is the merkle root information with trivial minimum constraints defined as: $g \geq 1$ and $p \geq 2$. For registration, W_p registers to G_g through generated metadata M_p and timestamp M_p^g recorded in request pool based on FCFS policy. In a particular game G_g, we consider two players P_a and P_b, with condition $a \neq b$, as opponents in G_g. Their combined metadata in G_g is represented as M_{ab}^g and is represented as follows:

$$M_{ab}^g = \left\{T_a^g, T_b^g, W_a^g, W_b^g, \text{ID}_{ab}^g\right\} \tag{2}$$

where T_a^g and T_b^g denotes the transactional entries of P_a and P_b, respectively, in G_g, W_a^g, and W_b^g denotes the respective wallets of the opponents for G_g, and ID_{ab}^g represents the combined ID of P_a and P_b in G_g. Based on M_{ab}^g, a common request R_{ab} for generation of $\mathbb{R}$ is entered in common request pool M_p^g as follows:

$$R_{ab} \leftarrow \left\{M_{ab}^g, N_{ab}, T_{ab}, B(\mathbb{R})\right\} \tag{3}$$

where N_{ab} is the required security level of the generated random number $\mathbb{R}$, T_{ab} is the common timestamp of request generation, and $B(\mathbb{R})$ is the sequence of random bits $\{b_i, b_{i+1}, \ldots, b_j\}$ from $\mathbb{R}$ to game G_g. A collection of such requests is stored in M_p^g for different associated p players mapped to different game-sets G.

To generate secure random number $\mathbb{R}$, we consider entity E_m, which consists of existing (old) miners in B denoted as $M_e = \{M_1, M_2, \ldots, M_e\}$ and new nodes proposed as miners $M_n = \{M_{e+1}, M_{e+2}, \ldots, M_n\}$ with constraints as follows:

$$\begin{aligned} C_1 &= e \geq 1 \\ C_2 &= n - (e+1) \geq 0. \end{aligned} \tag{4}$$

Then, we define A as aggregate set of all miner E_m in the system as follows:

$$\begin{aligned} A &= M_e + M_n \\ A &= \{M_1, M_2, \ldots, M_e, M_{e+1}, M_{e+2}, \ldots, M_{e+n}\}. \end{aligned} \tag{5}$$

As the arrival of new miners is nondeterministic and continuous so we define a miner pool M of size s. To prevent collusion attacks by dishonest miners, conditional probability $C_i = P(T_{b_i}|Y_{b_i})$ is computed for any ith miner entity E_m. Here, T_{b_i} denotes the true block proposals proposed by the ith miner validated in B and added as new blocks and Y_{b_i} denotes the total block proposals by ith miner. If $C_i > 0.51$, a boolean trust value T_i is mapped to 1 for ith miner and T_i is mapped to 0, otherwise. Based on C_i, a trust value $\{T_1, T_2, \ldots, T_{e+n}\}$ is assigned to each miner in A. Once M is full, then the aggregate set A is passed through a miner elimination process in which T_i is compared against a minimum threshold T_{min}. T_{min} is computed by measuring deviation ω of block additions of ith

miner from valid proposals by all other miners. From the total aggregate set A, we form a reduced miner set A_{red} by eliminating colluding miners A_{el}. The detailed process is explained in Section IV-C. A_{red} is denoted as follows:

$$|A_{\text{red}}| = |A| - |A_{el}|. \tag{6}$$

We compute a reduced set TR_k that denotes the trust value of k miners in A_{red} denoted as $\{T_1, T_2, \ldots, T_k\}$. Based on trust scores, A_{red} is divided into entities $\{A_c, A_g\}$, which include the group creators and value generators, respectively. As explained in Section III-A, metadata information M_g^p is collected by A_c for entity E_{GP} and the resultant is mapped with TR_k. Now, any k^{th} miner fetches TR_k with trust values from set $\{T_1, T_2, \ldots, T_k\}$ and matches the value against the measured trust deviation ω to validate block proposals. These rules are described as follows.

1) *Rule 1: If* $T_k \geq \omega$ and $k \in (e+n)$ then $A_g \leftarrow A_k$.
2) *Rule 2: Else-if* $T_k \geq 0$ then $A_g \leftarrow A_k$.

The mean deviation ω is computed as follows:

$$\omega = \sum_{i=1}^{e+n} T_i - \mu\left(\overline{\text{TR}_k}\right) \tag{7}$$

where $\overline{\text{TR}_k}$ is set of trust value of all miners in A_{red}. Consider q miners in A_c as $\{c_1, c_2, \ldots, c_q\}$ and s miners in A_g as $A_g = \{g_1, g_2, \ldots, g_s\}$. The entities A_c and A_g are subject to the constraints as follows:

$$C_3 : |q| + |s| \leq |e| + |n|. \tag{8}$$

The metadata collected from request pool are clustered into w groups by q miners in A_c denoted as GR_q and defined as follows:

$$GR_q = \{f_1, f_2, \ldots, f_w\}. \tag{9}$$

The numbers of total groups created are w. Every group w is assigned a generator A_g^w, to generate an independent value denoted by v_w. Timestamp of all v_w are recorded as t_w. Based on v_w and t_w, two vectors V and T are initialized as inputs to the random string generation algorithm, which is represented as follows:

$$\begin{aligned} V &= \{v_1, v_2, \ldots, v_w\} \\ T &= \{t_1, t_2, \ldots, t_w\}. \end{aligned} \tag{10}$$

$\{V, T\}$ generates a string of random sequence $\{b_0, b_1, \ldots, b_n\}$ as *FS*. *FS* is divided into subsequences $b_k, b_{k+1}, \ldots, b_l$ as oracles for generation of PSC denoted as $\{\text{PSC}_1, \text{PSC}_2, \ldots, \text{PSC}_k\}$ used to satisfy E_{GP} request from pool. Any k^{th} PSC is executed among E_{GP} and E_{GO} based on input oracle R_n. The list of symbols and their associated meanings are defined in Table III.

TABLE III
SYMBOLS AND ASSOCIATED MEANINGS

Symbol	Notations Used
B	Blockchain
p	number of players
g	number of games
P	Players
M	Miners
G	Games
W_p^g	Wallet of p^{th} player in g^{th} game
M_p^g	Metadata of p^{th} player in g^{th} game
Q	Queue of the requests for RNG algorithm
M_e	existing miners
M_n	new miners
A	Aggregation Of new and existing miners
TR_k	Trust value of k^{th} miner
A_{red}	Reduced set of miners
A_el	Set of Eliminated miners
ω	Threshold Trust Value
A_c	Set of Group Creator
A_g	Set of Value generator
GR	Group of requests
$\overline{TR_k}$	Trust values of all miners in A_{red}
V	Set of Value generated
T	Set of timestamp of Value submitted
RS	Randomly Generated String
$\mathbb{R}$	Random Number

IV. *SaNkhyA*: THE PROPOSED SCHEME

As discussed in Section III, we present the interaction sequences among different entities in the game G played between E_{GP} and E_{GO} as shown in Fig. 2. We consider that mapping for any pth player, presented as public wallet address to gth game. The wallet keys are independent of underlying BC, and thus can be referenced without access to chain structure and network connectivity, simplifying the overall complexity. Once the game G receives M_p, it generates a request to BC for generation of $\mathbb{R}$, that invokes the entities M_e and M_n. They generate the numbers based on PoV, and operate on reduced miner set A_{red}, by eliminating the colluding miners A_{el}. The request Q_r is generated as an asynchronous signal request, and miner entities compute the conditional probability $P(T_{b_i}|Y_{b_i})$, to ensure trust in the ecosystem. Once the random bit sequences are generated, they are communicated back to BC. The step is conformed as request satisfaction, which is a single-bit flag process. The game G decides the winner W based on execution of PSC, which are splitted into k PSC, based on randoms-splitter process. The game fairly selects the winner and presents an acknowledgement (ACK), or negative ACK (NAK) to E_{GP}. As the entire process is based on generation of random oracles for PSC, the inherent complexity of the scheme is proportional to the computational time required for generation of *FS*.

For entity E_{GP}, we consider two players P_a an P_b in game G. The players collectively places request for generation of secure random oracle $\mathbb{R}$ to request pool Q_r. Miner A proposes a novel *Miner_Selection_Algorithm* is proposed to create A_c and A_g. Then, based on received requests from A_c, a novel consensus procedure PoV is proposed so that A_{red} achieves a common truth to generate $\mathbb{R}$, and eliminate collusion attacks. Based on PoV, a random number generation algorithm is proposed to generate stream of bits in *FS*, and then splitter mechanism is proposed to generate subsequences for k different $\{\text{PSC}_1, \text{PSC}_2, \ldots, \text{PSC}_k\}$, based on winner of game G. Then, proposed k PSC are executed between E_{GP} and E_{GO} to process the transfer of funds securely. The details of these phases are presented as follows.

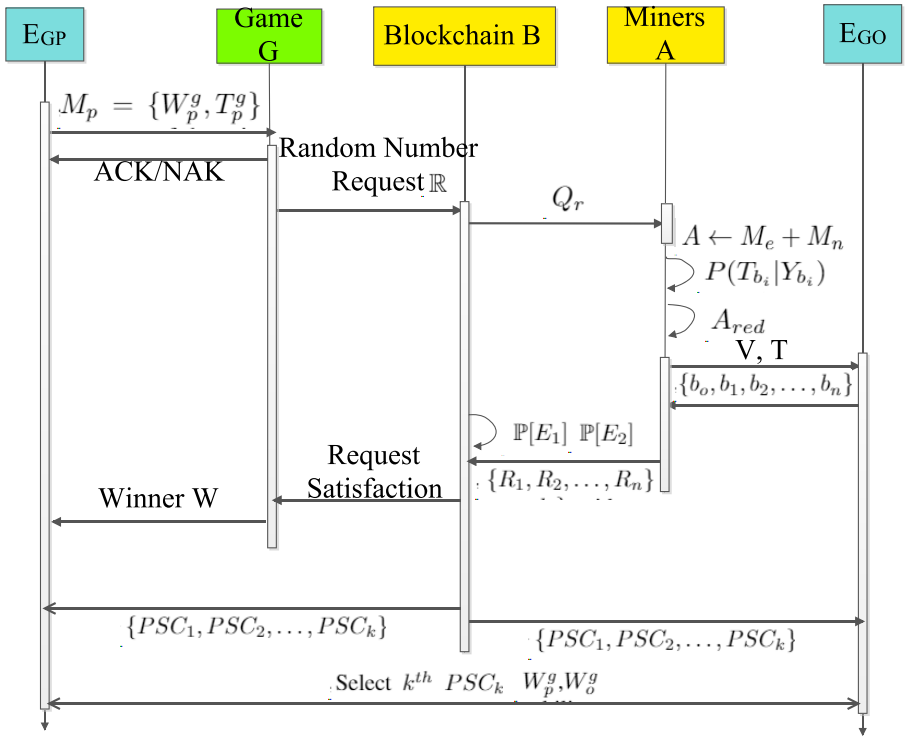


Fig. 2. *SaNkhyA*: interaction sequences among different entities.

Algorithm 1 SaNkhya: Mapping Requests to A_c and Value Generation by A_g

Input: GR_q,PK_g^c
Output: v_w,t_w
Initialization: $i = 1$,$message =$ "*OpenGroup*"
procedure MINER_SELECTION_ALGORITHM(GR_q)
  Broadcast(GR_q, M_p^g)
  $Q_{active} \leftarrow$ *Accept_Response*(A_g)
  $message =$ "*ClosedGroup*"
  Broadcast(GR_q, M_p^g)
  e : *map*(GR_q, A_g)
end procedure
procedure MINER_VALUE_GENERATION(e, PK_g^c)
  for i to $w-2$ **do**
    Extract (v_w,M_{ab}^g)
    $x \leftarrow$ *Hash* (T_a^g,W_b^g), $y \leftarrow$ *Hash*(ID_{ab}^g,PK_g^c), $z \leftarrow$ *Hash*(T_b^g,W_a^g)
    $x \leftarrow$*Hash*(x,y), $z \leftarrow$ *Hash*(y, z)
    Array_Values[i] $\leftarrow$ *Hash*(x,z)
  end for
  $v_w \leftarrow$ *Hash*(*Array_values*)
  $t_w \leftarrow$ *Block.timestamp*
end procedure

A. SaNkhyA: Group Creators A_c and Value Generators A_g

In this section, we propose an algorithm for miner selection and value generators A_g. Based on A_g, we propose a random number generation algorithm in which G_g sends a request for generation of $\mathbb{R}$ to the request pool M_p^g. The metadata M_p^g is clustered into w groups to form GR_q. Then, R_q is broadcasted to group creators A_g and a mapping e is formed to map any wth group details to any A_g following the FCFS policy. Public key of miner A_g is computed and passed as inputs to value generation algorithm denoted as PK_g^c.

Based on mapping e, w groups are considered and process is iterated $\forall w$ groups. In each group, a generator A_g^w is assigned independently. A_g^w generates an independent value, denoted as v_w at timestamp t_w. From v_w, metadata M_{ab}^g is extracted and is hashed and result is stored as output hashes x y, and z. x stores the hash of transactional entries of P_a with wallet of P_b, y stores hash of combined ID of P_a and P_b with public identifiers, and z stores the hash of transactional entries of P_b with a wallet of P_a. To map transactional entries with ID, the hashes are further hashed in a recursive binary tree fashion for $(w-2)$ inputs and extracted to form fixed output, stored in an array that stores hashes for all $(w-2)$ entries $\{v_1, v_2, \ldots, v_{w-2}\}$ at timestamps $\{t_1, t_2, \ldots, t_{w-2}\}$. The final hash is stored as root, or Merkle hash value denoted by v_w at timestamp t_w. Then these combined values are passed to the random number generation algorithm to produce the final random number $\mathbb{R}$. The details of the random number generation are given in Algorithm 1. As the algorithm processes w inputs as binary tree construction, the overall complexity of the algorithm is $O(w.GR_q)$.

B. PoV: Proof-of-Validity Consensus Mechanism

A_c collects requests from miner pool M_p^g and this request is passed to A_g selected based on FCFS policy. Consider A_g can create w groups denoted as $\{A_{g1}, A_{g2}, \ldots, A_{gw}\}$. Any group A_{gw} can have members $\{M_1, M_2, \ldots, M_k\}$, where $k < w$. Then, the following steps are performed to frame out the consensus.

1) *Agreement:* All k members in A_{gw} share their secret values $\{S_1, S_2, \ldots, S_k\}$ with each other, encrypted with

public key of remaining $k-1$ members, that ensures secrecy and confidentiality. The secret values S_k are then broadcasted by group creator M_c in A_{gw} within its group to ensure an agreement.

2) *Publish Step:* In next phase, encrypted secret value is broadcasted by different group creators $\{M_{c1}, M_{c2}, \ldots, M_{ck}\}$ within their group. All group creators M_c informs other members that they have completed the *Agreement* step in their group. This information exchange is termed as *publish* step. The published information is denoted as $\{M_{p1}, M_{p2}, \ldots, M_{pk}\}$.
3) *Miner Elimination:* In case any kth group creator M_{ck} did not publish M_{pk}, then the respective group is eliminated and the trust value TR_k for their group is decreased based on the given rule set.
   a) *Initialization: threshold* $= \mu(\mathrm{TR}_k)$.
   b) *Rule 1:* Set $\mathrm{TR}_k \leftarrow \omega$.
   c) *Rule 2:* If M_{pk} is published, then $\mathrm{TR}_k = \mathrm{TR}_k + \alpha$, else $\mathrm{TR}_k = \mathrm{TR}_k - \alpha$, where α is a scaling constant.
   d) *Rule 3:* If $\mathrm{TR}_k < \omega$, then eliminate group G_k.

The consensus agreement is subject to the following constraints:

$$C_1 : \exists G_k : (\forall M_{ck} : M_{ck} \rightarrow H(k))$$
$$C_2 : !\nexists G_k : (\forall M_{ck} : M_{ck} \rightarrow D(k)). \tag{11}$$

Constraint C_1 specifies that at least one group G_k exits that have all honest members M_{ck} denoted as $H(k)$. Constraint C_2 specifies that no group G_k such that all members M_{ck} are dishonest denoted by $D(k)$. Based on these constraints, two events E_1 and E_2 are represented as follows:

$$E_1 : A_j : \quad \forall M_j \rightarrow H(j)$$
$$E_2 : A_j : \exists M_j \wedge H(j) \tag{12}$$

where E_1 specifies that all value generators M_j are honest for any jth group and E_2 signify that at least one value generator M_j is honest in jth group, where $j \in \{1, 2, \ldots, e + n/k\}$. The relation among E_1 and E_2 is depicted as $X = \bigcup_{j=1}^{e+n/k} E_1$ and $Y = \bigcap_{j=1}^{e+n/k} E_2$. To reach to the consensus, value of k needs to be selected such that $\mathbb{P}\ |X \cap Y| \approx 1$. To achieve this, consider $k = \epsilon \ln t$ where $t = e + n$ and $\epsilon > 0$ is an arbitrary constant that determines probability of occurrence of events E_1 and E_2.

Then, consider the probability of independent malicious value generators as p in the jth group and also all M_{cj} are colluding in the group j. Then, $\mathbb{P}[E_1(j)] = (1-p)^k = (1-p)^{\epsilon \ln t}$. Finally

$$\mathbb{P}\big[E_1(j)\big] = t^{-\epsilon \ln(1-p)^{-1}}. \tag{13}$$

Based on $\mathbb{P}[E_1(j)]$, $\mathbb{P}[E_1]$ can be written as follows:

$$\begin{aligned}\mathbb{P}[E_1] &= 1 - \mathbb{P}\left[\bigcap_{j=1}^{t} E_1(j)\right] \\ &= 1 - \Big(1 - (1-p)^k\Big)^{t/k} \\ &= 1 - \Big(1 - t^{-\epsilon \ln(1-p)^{-1}}\Big)^{\frac{t}{\epsilon \ln t}} \\ &\approx 1 - \exp\left(\frac{-t^{1-\epsilon \ln(1-p)^{-1}}}{\epsilon \ln t}\right).\end{aligned} \tag{14}$$

The above expression value is close to 1 iff $\epsilon \ln(1-p)^{-}1 < 1$ and t is infinitely large value. Similarly, $\mathbb{P}[E_2(j)]$ can be defined as $\mathbb{P}[E_2(j)] = (1-p)^k = 1 - t^{-\epsilon \ln p^{-1}}$. $\mathbb{P}[E_2]$ can be computed as follows:

$$\begin{aligned}\mathbb{P}[E_2] &= \big(\mathbb{P}\big[E_2(j)\big]\big)^{t/k} \\ &= 1 - \Big(1 - p^k\Big)^{t/k} \\ &= 1 - \Big(1 - t^{-\epsilon \ln p^{-1}}\Big)^{\frac{t}{\epsilon \ln t}} \\ &\approx \exp\left(-\frac{t^{-\epsilon \ln p^{-1}+1}}{\epsilon \ln t}\right) \\ &\approx 1 - \frac{t^{-\epsilon \ln p^{-1}+1}}{\epsilon \ln t}.\end{aligned}$$

If $\epsilon \ln p^{-1} > 1$, then $\mathbb{P}[\mathbb{E}_{\not=}]$ is close to 1. So *epsilon* must satisfy this condition

$$\frac{1}{\ln p^{-1}} < \epsilon < \frac{1}{\ln(1-p)^{-1}}. \tag{15}$$

It happened, if $p < 0.5$. Thus, to achieve k close to 1 more than 50% of the miners in the system need to be honest $H(k)$. Based on the computation of $P[E_1]$, $P[E_2]$, and ϵ, value generators A_g^w update the TR_k values in A_{red} to update v_w. Based on the values of $\{v_1, v_2, \ldots, v_w\}$, vector V is updated at timestamps $\{t_1, t_2, \ldots, t_w\}$, respectively, to fed as inputs in generator of random number $\mathbb{R}$ to generate stream of random bit-sequences as *FS*.

C. Novel Method to Generate Random Numbers

Post generation of v_w, the string V is passed to a random number generation algorithm as depicted in Algorithm 2, with timestamp information T to generate *FS*. *FS* then goes through a random splitter procedure that splits the string into equal subsequences. The process is depicted in Fig. 3. To achieve this, we consider the input sequences $\{V_1, V_2, \ldots, V_w\}$ in block ranges of size from 0 to 255 bits. Any wth block V_w is divided into left and right sub-blocks, denoted as $L(V_w)$ and $R(V_w)$, respectively. $L(V_w)$ consists of 128 bits ranging from 0 to 127 and $R(V_w)$ is from 128 to 255 bits. At each round, $L(V_w)$ is utilized for key-generation K. Then, to add diffusion, for any kth block, an XOR operation is performed between $R(V_{k-1})$ and $L(V_k)$, respectively. The result of XOR is 128 bit output. The rightmost sub-block is discarded at each round. In total, w blocks are present as inputs. After the first XOR, $(w-1)$ 128-bit blocks are constructed. These blocks go through a permutation sequence for 128 bits. To add confusion, the permutation sequence is generated based on key values from the previous round. The 128 bits $(w-1)$ permuted blocks are then subdivided into 64 bit left and right sub-blocks. The process is iterative and the final block is permuted through a final permutation to generate a random string sequence of $\{b_0, b_1, \ldots, b_n\}$ bits, denoted as *FS*. From *FS*, the random sequence $\mathbb{R}$ is generated.

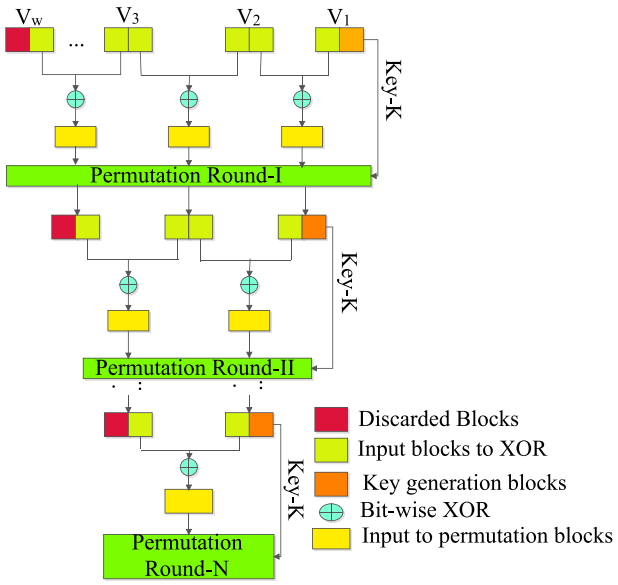


Fig. 3. *SaNkhyA*: random number generation.

The string FS is passed through random splitter to generate subsequences $\{b_k, b_{k+1}, \ldots, b_l\}$ as random oracles for execution of k PSC among E_{GP} and E_{GO}, based on winner of gth game in G. As the algorithm processes inputs as block sizes of 256 bits to produce $\mathbb{R}$, which are sorted the complexity is $O(k.w\log(w))$.

D. Implementation of Probabilistic Smart Contract

After $\mathbb{R}$ is generated, splitter procedure creates k subsequences as input oracles to k PSC, namely, $\{\text{PSC}_1, \text{PSC}_2, \ldots, \text{PSC}_k\}$. Any kth PSC is executed between winner E_{GP} an E_{GO} for fund transfer. Credit funds of winner of the game is stored by E_{GO} in variable *game.balance*. To select winner of a game G_g, consider two players a and b in G_g with wallets W_a an W_b, respectively. A random number in range $\{1, \ldots, 100\}$ is generated and stored in wallet addresses W_a and W_b, termed as x and y. Then, a lottery selection is executed to check whether the stored number x or y equals $\mathbb{R}$. If $\mathbb{R}$ equals x, a is declared as winner of game G_g and funds are transferred from *game.balance* to W_a. The transfer of funds is processed through PSC_k based on input oracle b_k. If $\mathbb{R}$ equals y, then b wins the game and *game.balance* is credited to W_b through PSC_k. The complexity depends on number of parallel users operational in the scheme. Consider k users in parallel, then the time complexity of execution of contract is $O(k.\mathbb{R})$ with interconnections as $O(k^2)$.

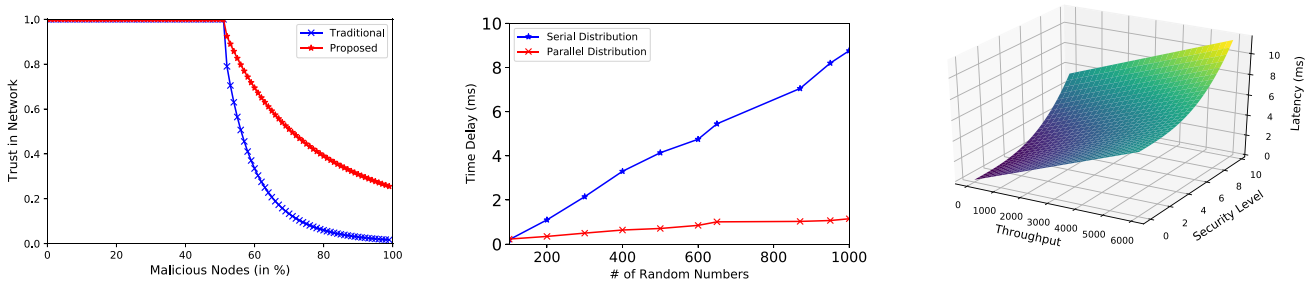
Algorithm 2 SaNkhya: Random Number Generation Algorithm

Input: GR_q,PK_g^c, v_w, t_w
Output: FS
Initialization: $i = 1, k = 1, count = 0$

procedure RANDOM_NUMBER_GENERATION
  $V_w, T_w \leftarrow MINER\_VALUE\_GENERATION(GR_q, PK_g^c)$
  Sort(V_w,T_w)
  for 1 to $w-1$ **do**
    $k \leftarrow V_1[0:127]$
    for every V_{w-1} and V_w, do
    $V_{w-1} \leftarrow V_{w-1}[128:255] \oplus V_w[0:127]$
    for i to length(K) **do**
      if mark(K_i)==False **then**
        if K_i==1 **then**
          for j=i+1 to length(K) **do**
            if K_j==1 **then**
              swap(V_w^i,V_w^j)
              $mark(K_i) \leftarrow True$
              $mark(K_j) \leftarrow True$
              GOTO step 7
            end if
          end for
        else
          for j=i+1 to length(K) **do**
            if K_j==0 **then**
              swap(V_w^i,V_w^j)
              $mark(K_i) \leftarrow True$
              $mark(K_j) \leftarrow True$
              GOTO step 7
            end if
          end for
        end if
      else
        GOTO step 7
      end if
    end for
  end for
  $\mathbb{R} \leftarrow FS \leftarrow V_w$
end procedure
procedure RANDOM_NUMBER_SPLITTER($\mathbb{R}$)
  for $i = 0$ to w **do**
    $l \leftarrow \mathbb{R}.length$
    for $j = k$ to l **do**
      $n \leftarrow r_j^i$
      $Count = Count + n$
      if $Count \leq size(w)$ **then**
        ($x \leftarrow$ Left_shift($\mathbb{R}$, n)
        Pass x to r_j^i
      end if
    end for
  end for
  $Q \leftarrow Purge(\mathbb{R})$
end procedure

V. Performance Evaluation

In this section, we evaluate the performance of the proposed scheme *SaNkhyA* against conventional approaches. For trust values TR_k, we compare the proposed scheme employing miner selection scheme, depicted in Section IV-B against existing nonminer selection schemes [6], [7]. Then, *SaNkhyA* is compared for parameters-latency creation in block generation, and scalability of mined transactions. For block generation, we compare the proposed work against existing serial schemes in [6], [9], and [11] and scalability of mined transactions is compared against nonminer selection schemes in [6] and [7]. Then, we evaluate throughput-latency trade-offs for *SaNkhyA* and compare the proposed scheme for delay in random number generation against traditional approaches in [7], [8], and [14]. For block-convergence time, we compare the proposed consensus PoV against traditional consensus schemes.



(a) (b) (c)

Fig. 4. *SaNkhyA*: impact of collusions and security of random number generation. (a) Impact on trust with measured collusions. (b) Latency comparison in block creation. (c) Security-latency tradeoff in *SaNkhyA*.

TABLE IV
SIMULATION PARAMETERS

Parameters	Values
G: Total Number Of Games	30
E_{GP}: Total Number Of Game Players	150
E_{GO}: Total Number of Game Owners	50
M_p: Meta-data of mapping request	50
M_e: Existing miners	10
M_n: New miners	10
$\mu(TR_k)$: Minimum threshold for trust	0.51
ω: Cumulative threshold trust value	0.29
FS: Final string (bit-length)	1-10000.
PSC: Probabilistic Contracts-set	40
M: Mined Transactions	1-6000
r: SC wallet random numbers for P_a and P_b	1-100

Algorithm 3 SaNkhya: Smart Contract Algorithm

Input: W_a,W_b
Output: $\mathbb{R}$
Initialization: $is = 1$,$es = 100$

$x \leftarrow select\_num(W_a, is, es)$
$y \leftarrow select\_num(W_b, is, es)$
$B_x \leftarrow lottery\_selection(W_a, x)$
$B_y \leftarrow lottery\_selection(W_b, y)$
$game.balance = B_x + B_y$
$\mathbb{R} = RANDOM\_VALUE\_GENERATION()$
if $x == \mathbb{R}$ **then**
    $W_a += game.balance$
    $game.balance = 0$
else if $y == \mathbb{R}$ **then**
    $W_b += game.balance$
    $game.balance = 0$
end if

A. Experimental Setup and Simulation Parameters

For experimental setup, we consider the formulation of PSC between E_{GP} and E_{GO} through using Remix IDE and execution through Solidity v0.5.11. The PSC are deployed on BC using the *Truffle* suite with npm v5.1.0. For generating plots, MATLAB Online R2019b v 9.7 is used. The details of the simulation parameters for the experimental setup are presented in Table IV.

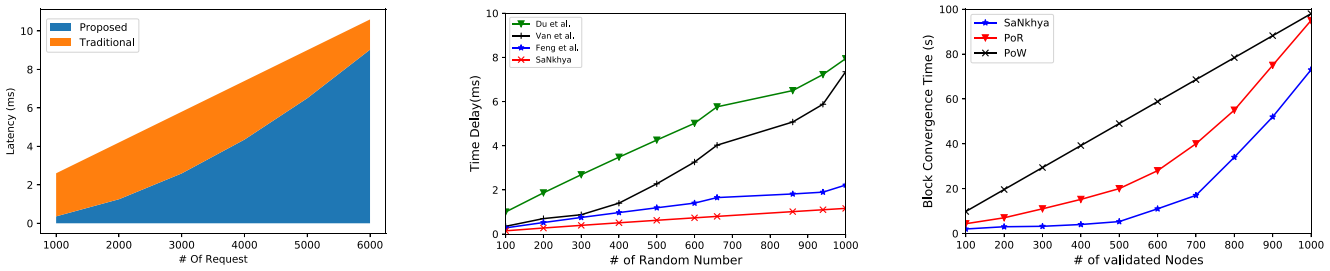
B. Simulation Results

1) Impact of Consensus and Security Level of Random Numbers: Based on the trust values of A_{red}, a consensus algorithm PoV is proposed to weed out colluding miners from M_{ck} at defined threshold $\mu(\text{TR}_k)$. For the same, Fig. 4(a) considers a mapping of TR_k to event $P[E_1]$. As evident from the graph, the trust drops drastically when colluding nodes are more than 50% in nonminer schemes, but the effect is lower in the proposed scheme. At 85% collusion, $P[E_1]$, TR_k is close to 0.38, indicating that the proposed scheme effectively discards M_{ck} in case M_{pk} broadcast is not reached. The reason for the same is improved correct oral propagation updates through honest miners.

Fig. 4(b) evaluates the time delay in generation of $\mathbb{R}$ with parallel block processing as indicated in Algorithm 2, against conventional serial approaches. As the proposed scheme processes block sequences $\{V_1, V_2, \ldots, V_w\}$ in parallel with any kth block V_k divided into $L(V_k)$ and $R(V_{k-1})$, processing delay is less compared to serial generation of blocks where any kth block input is processed when output of $(k-1)$th block is present. The average block processing delay is 1.3 s in *SaNkhyA* compared to serial approaches with an average process time of 5.6 s for 1000 blocks.

Fig. 4(c) depicts the relationship between the throughput of B at the desired security levels in the range of 0–10 and obtained latency to achieve the desired security. Any game G_g demands low throughput η with low-security level, then the latency Δ is less. If G_g demands a higher security level, then the throughput and latency increase gradually. This is evident from the fact that as the security level is increased, then the diffusion in block structure V_k increases and key size K also increases with more rounds w to generate $\mathbb{R}$. To support high encryption structures, keeping η as constant and Δ increases. IF η is made variable, then Δ is proportional to the security of $\mathbb{R}$ generation. Hence, processing applications face a security-latency tradeoff based on selected η as throughput.

2) PoV and Efficiency of Random Number Generation of SaNkhyA Against Conventional Approaches: Fig. 5(a) highlights the improved latency of *SaNkhyA* against nonminer selection schemes. As from A_{red}, based on deviation μ, dishonest miners are eliminated, honest miners are proposed higher incentives. Also, due to parallel block generation, bit-sequences $\{b_0, b_1, \ldots, b_n\}$ of generated $\mathbb{R}$ are decomposed into subsequences b_k in less-time compared to traditional nonminer selection approaches. Thus, for any G_g, input oracles to



(a) (b) (c)

Fig. 5. *SaNkhyA*: scalability and PoV measurement against conventional approaches. (a) Scalability of mined transactions. (b) Delay in random number generation. (c) PoV: improved BCT.

PSC$_k$ is presented with lower latency. Hence, more PSC are added per quantum of time, that leads to improved scalability.

Fig. 5(b) shows a significant improvement in measured average delay of *SaNkhyA* against other state-of-the art approaches [7], [14], [21]. This is due to the fact of parallel block creation in generation of $\mathbb{R}$.

Fig. 5(c) defines parameter block convergence time (BCT) to measure the improvements in proposed consensus mechanism PoV. BCT is defined as the time required by all group creators $\{M_{c1}, M_{c2}, \ldots, M_{ck}\}$ to perform *publish* step denoted as $\{M_{p1}, M_{p2}, \ldots, M_{pk}\}$. If *publish* step is unconfirmed, and M_{ck} is eliminated, then the scheme converges to common truth due to correct oral updates against traditional consensus mechanisms. However, as more players E_{GP} are added to the scheme, the convergence rate decreases as block validation increases and approaches linearity with increases in added blocks.

C. Functionalities of Probabilistic Smart Contracts

As shown in Fig. 6, secure PSC functionalities are designed between E_{GP} and E_{GO} for transfer of funds from *game.balance*, based on generated inputs PSC$_k$ from random oracle $\mathbb{R}$. Any E_{GP} registers to game G_g as shown in Fig. 6(a). Post registration, W_g^p is added to M_p and player p makes a request for generation of $\mathbb{R}$. These requests are collected in request pool from all players as M_p^g. The request for generation of $\mathbb{R}$ from M_p^g is depicted in Fig. 6(b). The call invokes Algorithm 2, and produces output string *FS*. Then, k subsequences are generated as input oracles for PSC$_k$ that selects game winner W among players P_a and P_b. Selection of W is depicted in Fig. 6(c). The transfer of funds is processed from *game.balance* and is credited to wallet of W and these steps are shown in Fig. 6(d).

D. Security Evaluation

This section evaluates the overall computation cost (CC) and communication cost (CCM) of *SaNkhyA* based on identifiers selected from [22]. The details are presented as follows.

1) Formal Verification: The section presents the formal verification of PSC in *SaNkhyA* for security flaws. Smart contracts are vulnerable to various attack vectors like transaction origin, re-entrant problem, order dependence of contract evaluations, gas attacks, and timestamp dependencies. Thus, it is

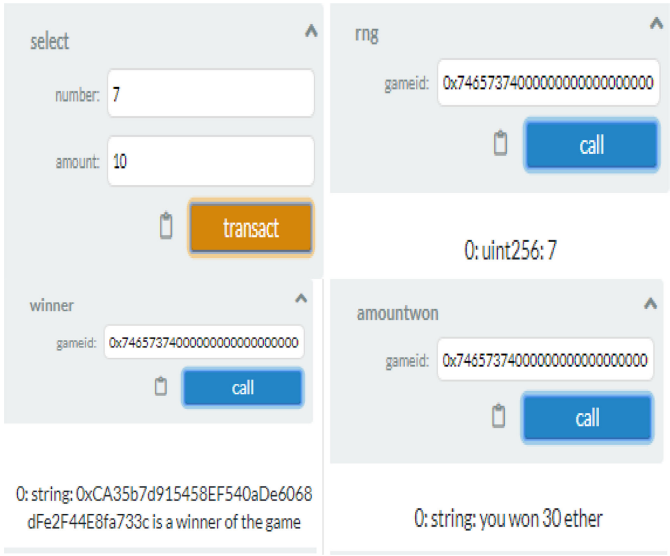


Fig. 6. Proposed SC functionalities in *SaNkhyA* scheme. (a) Select number. (b) Ask for a random number. (c) Winner. (d) Amount of money won.

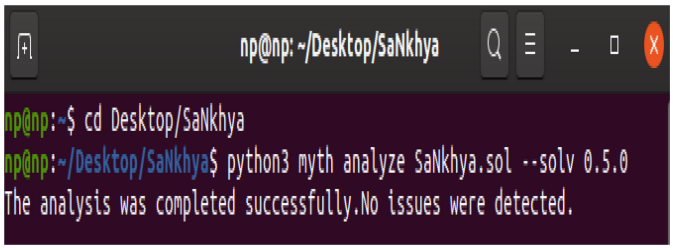


Fig. 7. Formal security verification of *SaNkhyA*.

imperative to verify the security validation of the proposed contract between the players P_p in the game G_g in the scheme before deploying the contract. *Mythril* performs security analysis techniques on contracts, for instance, taint analysis, control flow checks, reordering checks, and transactional flows. The contract file *SaNkhya.sol* is passed and no issues were detected as shown in Fig. 7.

2) Computation and Communication Cost: To compute CC, we first evaluate Algorithm 1. Metadata M_p^g is assigned to A_c. The process consists of asymmetric encryption and hash operations. Considering 10 users for broadcast, the cost is $10 * (0.0056 + 0.00032 + 0.00032) \approx 0.00624$ s. A_c performs a mapping A_g and the mapping cost is 0.0032 s. Then, the

TABLE V
COMPARISON OF OVERALL COMPUTATION (CC) AND COMMUNICATION COST (CCM) AGAINST EXISTING SCHEMES

Scheme	CC	CCM	ME
Odelu *et al.* [23]	$7E_{asym} + 12H_m + 2T_{pair} \approx 505.72ms$	240 bytes	3
Kabra *et al.* [16]	$2E_{asym} + 1T_{signgen} + 2T_{signver} + H_m \approx 192.14$ ms	203 bytes	4
Proposed *SaNkhyA*	$7H_m + 3nonce + 3E_{asym} \approx 28.48ms$	101 bytes	4

E_{asym}: Asymmetric encryption cost; H_m: Hash output cost; T_{pair}: Bilinear pairing cost; E_{sym}: Symmetric encryption cost; T_i : Transaction append cost; S_m: Signing cost; V_m: Verification cost; $T_{signgen}$: Signature generation cost;$T_{signver}$: Signature verification cost, $nonce$: Time-stamp cost; $merkle - root$: timestamp cost to refer genesis block hash; T_{append}: Cost of appending blocks to chain. .

TABLE VI
COMPARATIVE ANALYSIS WITH EXISTING SCHEMES

Parameters	Chatterjee *et al.* [6]	Nguyen-Van *et al.*[8]	Du *et al.*[7]	Feng *et al.* [14]	Proposed *SaNkhyA*
A1	✓	✓	✗	✓	✓
A2	✓	✓	✓	✓	✓
A3	✓	✓	-	✓	✓
A4	✗	✗	✓	✓	✓
A5	✗	✗	-	✗	✓

A1: Unpredictability; A2: Consensus; A3: Security; A4: Decentralised; A5: Privacy ; ✓ shows parameter is present; ✗ shows parameter is absent; & - shows parameter is not considered .

procedure *MINER_VALUE_GENERATION* contains a group request GR_q using public key PK_g^c. Total 6 hash operations are present and end block is appended in chain. Thus, the overall cost is $6*(0.00032) + 0.00032 + 0.00032 \approx 0.00256$ s. The total cost of Algorithm 1 is $0.00624 + 0.00256 + 0.00032 \approx 0.00912$ s. Algorithm 2 computes $\mathbb{R}$. It consists of one asymmetric encryption of V_w and 1 hash operation, thus the overall cost is $0.00032 + 0.00056 \approx 0.00816$ s. Algorithm 3 builds a PSC between P_a and P_b. It consists of two asymmetric encryption of wallet W_a and W_b, and 1 nonce identifier. The overall cost is $0.0056*2+0.00032 \approx 0.0112$ s. Thus, the overall CC of *SaNkhyA* is $0.00912 + 0.00816 + 0.0112 \approx 0.02848$ s or 28.48 milliseconds.

CCM is evaluated in same pattern of CC. Algorithm 1 consists of broadcast information, of 1 bits. Considering 10 users, the broadcast takes 10 bits. Mapping e is 1 bit. Hash identity of ID_{ab}^g is 160 bits. Block timestamp append t_w is 160 bits. Thus, total CCM of Algorithm 1 is $10 + 1 + 160 + 160 \approx 331$ bits. Algorithm 2 consists of XOR operation of 128 bits, boolean mark operation of 1 bit, and public-key-operation PK_g^c of 32 bits. Thus, overall CCM of Algorithm 2 is $128 + 1 + 32 \approx 161$ bits. Algorithm 3 consists of 2 wallet information exchanges, each of 160 bits hash output. *game.balance* checking condition requires 1 bit for flow condition. Thus, CCM for Algorithm 3 is $160 + 160 + 1 \approx 321$ bits. Thus, the overall CCM of *SaNkhyA* is $331 + 161 + 321 \approx 813$ bits or 101 bytes. Table V shows the overall comparison of CC, CCM, and several messages exchange (ME) against existing schemes.

E. Comparative Analysis

The proposed scheme *SaNkhyA* is compared against traditional schemes. Table VI presents the comparative analysis against conventional security models on BC. Results indicate the proposed scheme outperforms other state-of-the-art approaches against chosen parameters.

VI. CONCLUSION

In modern decentralized IoT networks, randomization in input oracles is critical for secured sensor-based exchange over open channels. Also, a randomized nonce is required in transactional tradings pertaining to financial institutions and windfall games. It allows nondeterminism in input oracles to PSC and leverages an efficient ecosystem to tackle colluding dishonest entities. In a similar direction, the BC-envisioned scheme *SaNkhyA* is proposed for the IoT-enabled PSC ecosystem that ensures fairness and transparency in game-events. The scheme exploits a miner selection algorithm that outputs an unbiased independent value. The value is then fed to the proposed consensus scheme PoV that formulates a miner-elimination process through a stable agreement scheme among all miners. The scheme weeds out colluding miners from the system. The step ensures that block additions are fair and transparent. The honest miners then agree on the generation of random oracle $\mathbb{R}$ as a sequence of stream bits, that are passed through a splitting mechanism, to generate secure random oracles to different PSC. The obtained results indicate the efficacy of the proposed scheme. In the future, we will propose a light-weight signing mechanism for E_{GP} and E_{GO} using lattices to secure the proposed scheme against colluding miners that can induce quantum attacks in the system.

REFERENCES

[1] H. Yao, T. Mai, J. Wang, Z. Ji, C. Jiang, and Y. Qian, "Resource trading in blockchain-based industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3602–3609, Jun. 2019.

[2] P. Bhattacharya, A. Singh, A. Kumar, A. K. Tiwari, and R. Srivastava, "Comparative study for proposed algorithm for all-optical network with negative acknowledgement (AO-NACK)," in *Proc. 7th Int. Conf. Comput. Commun. Technol. (ICCCT)*, New York, NY, USA, 2017, pp. 47–51.

[3] N. Grech, M. Kong, A. Jurisevic, L. Brent, B. Scholz, and Y. Smaragdakis, "MadMax: Surviving out-of-gas conditions in Ethereum smart contracts," in *Proc. ACM Program. Lang.*, vol. 2, Oct. 2018, pp. 1–27.

[4] N. Atzei, M. Bartoletti, T. Cimoli, S. Lande, and R. Zunino, "SoK: Unraveling bitcoin smart contracts," in *Principles of Security and Trust* (Lecture Notes in Computer Science), vol. 10804, L. Bauer and R. Küsters, Eds. Cham, Switzerland: Springer, 2018, pp. 217–242. doi: 10.1007/978-3-319-89722-6_9.

[5] I. Nikoli, A. Kolluri, I. Sergey, P. Saxena, and A. Hobor, "Finding the greedy, prodigal, and suicidal contracts at scale," in *Proc. 34th Annu. Comput. Security Appl. Conf. (ACSAC)*, New York, NY, USA, 2018, pp. 653–663.

[6] K. Chatterjee, A. K. Goharshady, and A. Pourdamghani, "Probabilistic smart contracts: Secure randomness on the blockchain," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, Seoul, South Korea, May 2019, pp. 403–412.

[7] M. Du, Q. Chen, L. Liu, and X. Ma, "A blockchain-based random number generation algorithm and the application in blockchain games," in *Proc. IEEE Int. Conf. Syst. Man Cybern. (SMC)*, Bari, Italy, Oct. 2019, pp. 3498–3503.

[8] T. Nguyen Van *et al.*, "Scalable distributed random number generation based on homomorphic encryption," in *Proc. Blockchain*, Jul. 2019, pp. 572–579.
[9] J. Choi, W. Shin, J. Kim, and K. Kim, "Random seed generation for IoT key generation and key management system using blockchain," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Barcelona, Spainm 2020, pp. 663–665.
[10] J. Li, Z. Zhang, and M. Li, "BANFEL: A blockchain based smart contract for fair and efficient lottery scheme," in *Proc. IEEE Conf. Depend. Secure Comput. (DSC)*, Hangzhou, China, Nov. 2019, pp. 1–8.
[11] Z. Wang, H. Yu, Z. Zhang, J. Piao, and J. Liu, "Ecdsa weak randomness in bitcoin," *Future Gener. Comput. Syst.*, vol. 102, pp. 507–513, Jan. 2020.
[12] P. Datta, "Constrained pseudorandom functions from functional encryption," *Theor. Comput. Sci.*, vol. 809, pp. 137–170, Feb. 2020.
[13] Y. Ehara and M. Tada, "How to generate transparent random numbers using blockchain," in *Proc. Int. Symp. Inf. Theory Appl. (ISITA)*, Singapore, Oct. 2018, pp. 169–173.
[14] J. Feng, X. Zhao, K. Chen, F. Zhao, and G. Zhang, "Towards random-honest miners selection and multi-blocks creation: Proof-of-negotiation consensus mechanism in blockchain networks," *Future Gener. Comput. Syst.*, vol. 105, pp. 248–258, Apr. 2020.
[15] M. R. Dorsala, V. Sastry, and S. Chapram, "Fair payments for verifiable cloud services using smart contracts," *Comput. Security*, vol. 90, pp. 1–19, Mar. 2020.
[16] N. Kabra, P. Bhattacharya, S. Tanwar, and S. Tyagi, "MudraChain: Blockchain-based framework for automated cheque clearance in financial institutions," *Future Gener. Comput. Syst.*, vol. 102, pp. 574–587, Jan. 2020.
[17] P. Bhattacharya, S. Tanwar, R. Shah, and A. Ladha, "Mobile edge computing-enabled blockchain framework—A survey," in *Proc. ICRIC*, 2020, pp. 797–809.
[18] U. Bodkhe, P. Bhattacharya, S. Tanwar, S. Tyagi, N. Kumar, and M. S. Obaidat, "BloHosT: Blockchain enabled smart tourism and hospitality management," in *Proc. Int. Conf. Comput. Inf. Telecommun. Syst. (CITS)*, Aug. 2019, pp. 1–5.
[19] J. Choi, "On throughput of compressive random access for one short message delivery in IoT," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3499–3508, Apr. 2020.
[20] *Solidity Pitfalls: Random Number Generation for Ethereum*. Accessed: Jun. 4, 2018. [Online]. Available : https://www.sitepoint.com/solidity-pitfalls-random-number-generation-for-ethereum/
[21] T. Nguyen-Van *et al.*, "A system for scalable decentralized random number generation," in *Proc. IEEE 23rd Int. Enterprise Distrib. Object Comput. Workshop (EDOCW)*, Paris, France, Oct. 2019, pp. 100–103.
[22] S. B. Patel, P. Bhattacharya, S. Tanwar, and N. Kumar, "KiRTi: A blockchain-based credit recommender system for financial institutions," *IEEE Trans. Netw. Sci. Eng.*, early access, Jun. 29, 2020, doi: 10.1109/TNSE.2020.3005678.
[23] V. Odelu, A. K. Das, M. Wazid, and M. Conti, "Provably secure authenticated key agreement scheme for smart grid," *IEEE Trans. Smart Grid*, vol. 9, no. 3, pp. 1900–1910, May 2018.

Nikunjkumar Sureshbhai Patel is currently pursuing the graduation degree with Nirma University, Ahmedabad, India.

His research interest lies in the area of integration of blockchain technology with IoT with a special focus on securing financial transactions.

Pronaya Bhattacharya received the B.Tech degree from Uttar Pradesh Technical University, Lucknow, India, in 2008, and the M.Tech degree from IPM Institute, Karnataka State University, Mysore, India, in 2012. He is currently pursuing the Ph.D. degree with specialization in optical communications from Dr. A. P. J Abdul Kalam Technical University, Lucknow, India.

He is employed as an Assistant Professor with the Computer Science and Engineering Department, Institute of Technology, Nirma University, Ahmedabad, India. He has over eight years of teaching experience. He has authored or co-authored 35 research papers in leading SCI journals and top core IEEE ComSoc conferences. Some of his top findings are published in IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, *Transactions on Emerging Telecommunications Technologies* (Wiley), *Future Generation Computer Systems* (Elsevier), *Journal of Engineering Research* (Kuwait University), ACM-MobiCom, IEEE-InfoCom, IEEE-ICC, and IEEE-CITS. His research interests include optical switching, high-performance networking, blockchain, IoT, and deep learning.

Mr. Bhattacharya is awarded the best paper award in Springer ICRIC-2019 and COMS2-2021. He is working as a Reviewer of reputed SCI journals IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE NETWORK, IEEE-ACCESS, *Transactions on Emerging Telecommunications Technologies* (Wiley), *Scandinavian Political Studies* (Wiley), *International Journal of Communication Systems* (Wiley), *Optical Switching and Networking* (Elsevier), *Multimedia Tools and Applications* (Springer), *Wireless Personal Communications* (Springer), and *Journal of Engineering Research* (Kuwait University). He has also been appointed as the Session Chair in IC4S-2019, IC4S-2020, and IICT-2020, organized by Springer. He is a Lifetime Member of professional societies like ISTE and IAENG.

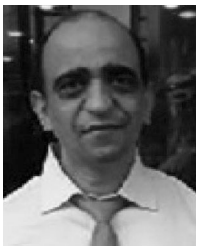
Shivani Bharatbhai Patel is currently pursuing the graduation degree with Nirma University, Ahmedabad, India.

Her research interests lie in the area of blockchain technology, deep learning, and IoT to solve real-world problems.

Sudeep Tanwar (Senior Member, IEEE) received the B.Tech degree from Kurukshetra University, Kurukshetra, India, in 2002, M.Tech degree (Honor's) from Guru Gobind Singh Indraprastha University, Delhi, India, in 2009, and the Ph.D. degree with specialization in wireless sensor network from Mewar University, Chittorgarh, India, in 2016.

He is currently an Associate Professor with the Computer Science and Engineering Department, Institute of Technology, Nirma University, Ahmedabad, India. He is a Visiting Professor with Jan Wyzykowski University, Polkowice, Poland, and the University of Pitesti Piteşti, Piteşti, Romania. He has authored 2 books and edited 13 books, over 200 technical papers, including top journals and top conferences, such as IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE WIRELESS COMMUNICATIONS, *Networks*, ICC, GLOBECOM, and INFOCOM. His h-index is 36. His research interests include blockchain technology, wireless sensor networks, fog computing, smart grid, and IoT. He initiated the research field of blockchain technology adoption in various verticals in 2017. He actively serves his research communities in various roles.

Dr. Tanwar has been awarded the best research paper awards from IEEE GLOBECOM 2018, IEEE ICC 2019, and SpringerICRIC-2019. He is currently serving the editorial boards of *Physical Communication*, *Computer Communications*, *International Journal of Communication System*, and *Security and Privacy*. He has served many international conferences as a member of the organizing committee, such as the Publication Chair for FTNCT-2020, ICCIC 2020, WiMob2019, the Member of the advisory board for ICACCT-2021, ICACI 2020, the Workshop Co-Chair for CIS 2021, and the General Chair for IC4S 2019, 2020, and ICCSDF 2020. He is a final voting member for IEEE ComSoc Tactile Internet Committee in 2020. He is a Senior Member of CSI, IAENG, ISTE, and CSTA, and the member of the Technical Committee on Tactile Internet of IEEE Communication Society



Neeraj Kumar (Senior Member, IEEE) received the Ph.D. degree in CSE from Shri Mata Vaishno Devi University, Katra, India, in 2009.

He is currently working as a Full Professor with the Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Deemed to be University, Patiala, India, and also with also with King Abdulaziz University, Jeddah, Saudi Arabia, and also with the Department of Computer Science and Information Engineering, Asia University, Taichung City, Taiwan. He is a Visiting Professor with Coventry University. He has published over 300 technical research papers in leading journals and conferences from IEEE, Elsevier, Springer, and John Wiley. Some of his research findings are published in top-cited journals, such as IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON CONSUMER ELECTRONICS, IEEE NETWORKING, IEEE COMMUNICATION, IEEE WIRELESS COMMUNICATIONS, IEEE INTERNET OF THINGS JOURNAL, IEEE SYSTEMS JOURNAL, *Future Generation Computer Systems*, *Journal of Network and Computer Applications*, and *Computer Communications*. He has guided many Ph.D. and M.E./M.Tech. His research is supported by fundings from the Tata Consultancy Service, the Council of Scientific and Industrial Research, and the Department of Science and Technology.

Dr. Kumar has been awarded the best research paper awards from IEEE ICC 2018 and IEEE Systems Journal 2018. He is leading the research group Sustainable Practices for the Internet of Energy and Security where group members are working on the latest cutting-edge technologies. He is a TPC member and reviewer of many international conferences across the globe.

Houbing Song (Senior Member, IEEE) received the M.S. degree in civil engineering from The University of Texas at El Paso, El Paso, TX, USA, in 2016, and the Ph.D. degree in electrical engineering from the University of Virginia, Charlottesville, VA, USA, in August 2012.

He was an Engineering Research Associate with the Texas A&M Transportation Institute, Bryan, TX, USA, in 2007. From 2012 to 2017, he was with the Faculty of West Virginia University, Morgantown, WV, USA. In 2017, he joined the Department of Electrical, Computer, Software, and Systems Engineering, Embry-Riddle Aeronautical University, Daytona Beach, FL, USA, where he is currently an Assistant Professor and the Director of the Security and Optimization for Networked Global Laboratory. He has authored over 100 articles. His research interests include cyber-physical systems, cybersecurity and privacy, the Internet of Things, edge computing, big data analytics, unmanned aircraft systems, connected vehicles, smart and connected health, and wireless communications, and networking.

Dr. Song was a recipient of the Navigation and Surveillance Technologies (ICNS) Conference, the very first recipient of the Golden Bear Scholar Award, the Highest Campus-Wide Recognition for Research Excellence at the West Virginia University Institute of Technology (WVU Tech), in 2016, the prestigious Air Force Research Laboratory's Information Directorate (AFRL/RI) Visiting Faculty Research Fellowship, in 2018, and the Best Paper Award from 2019 Integrated Communication. He has served as an Associate Technical Editor for the *IEEE Communications Magazine*. He has served as a Guest Editor for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, the IEEE INTERNET OF THINGS JOURNAL, the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, and the IEEE NETWORK. He is the Editor of six books, including *Big Data Analytics for Cyber-Physical Systems: Machine Learning for the Internet of Things* (Elsevier, 2019), *Smart Cities: Foundations, Principles, and Applications* (Hoboken, NJ, USA: Wiley, 2017), *Security and Privacy in Cyber-Physical Systems: Foundations, Principles, and Applications* (Chichester, U.K.: Wiley-IEEE Press, 2017), *Cyber-Physical Systems: Foundations, Principles, and Applications* (Boston, MA, USA: Academic Press, 2016), and *Industrial Internet of Things: Cybermanufacturing Systems* (Cham, Switzerland: Springer, 2016). He is a Senior Member of ACM.