Future Generation Computer Systems 124 (2021) 68–79

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A blockchain-based trust system for decentralised applications: When trustless needs trust

Nguyen Truong [a,*], Gyu Myoung Lee [b], Kai Sun [a], Florian Guitton [a], YiKe Guo [a,c]

[a] Data Science Institute, South Kensington Campus, Imperial College London, London SW7 2AZ, United Kingdom
[b] Department of Computer Science, Liverpool John Moores University, Liverpool L3 3AF, United Kingdom
[c] Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong

ARTICLE INFO

Article history:
Received 26 January 2021
Received in revised form 7 April 2021
Accepted 20 May 2021
Available online 24 May 2021

Keywords:
Blockchain
DApps
Decentralised ecosystem
Reputation
Trust system

ABSTRACT

Blockchain technology has been envisaged to commence an era of decentralised applications and services (DApps) without the need for a trusted intermediary. Such DApps open a marketplace in which services are delivered to end-users by contributors which are then incentivised by cryptocurrencies in an automated, peer-to-peer, and trustless fashion. However, blockchain, consolidated by smart contracts, only ensures on-chain data security, autonomy and integrity of the business logic execution defined in smart contracts. It cannot guarantee the quality of service of DApps, which entirely depends on the services' performance. Thus, there is a critical need for a trust system to reduce the risk of dealing with fraudulent counterparts in a blockchain network. These reasons motivate us to develop a fully decentralised trust framework deployed on top of a blockchain platform, operating along with DApps in the marketplace to demoralise deceptive entities while encouraging trustworthy ones. The trust system works as an underlying decentralised service providing a feedback mechanism for end-users and maintaining trust relationships among them in the ecosystem accordingly. We believe this research fortifies the DApps ecosystem by introducing an universal trust middleware for DApps as well as shedding light on the implementation of a decentralised trust system.

1. Introduction

The turn of the last decade brought us to the disruptive Blockchain technology (BC) that provides a trusted infrastructure for enabling a variety of decentralised applications and services (DApps) without the need for an intermediary. To actualise this vision, Smart Contracts (SCs) technology is consolidated into the BC-based infrastructure: SCs are programmed to perform services' business logic, compiled into byte-code, and deployed onto a BC platform (i.e., replicated into full-nodes in the platform) so that a user can create transactions to execute the business logic implemented in the SCs in a decentralised fashion [1]. This infrastructural BC platform offers some advanced features including immutability, transparency, trace-ability, and autonomy that are promising to effectively implement plentiful DApps from financial services (i.e., cryptocurrencies trading) to numerous services such as digital asset management [2], provenance tracking in logistics and supply-chain [3,4], and data sharing and processing in the Internet of Things (IoT) [5,6].

Indeed, various DApps have already been developed and employed into the real-world. For instance, there are over 4000 DApps deployed on top of the Ethereum, Tron, and EOS platforms, serving about 150*k* active users daily in 2019.[1] This is a considerable ecosystem and a huge decentralised peer-to-peer (P2P) marketplace. Although there are numerous challenges due to the limitation of the current BC technology hindering the advancement of DApps, we believe that *"everything that can be decentralized, will be decentralized" - David A. Johnston.*[2] The DApps ecosystem is just in its preliminary state and will be the future of the next-generation Internet.

1.1. Features of DApps

There are different perspectives of DApps definition and system development among the cryptocurrency space. Nonetheless, mutual perceptions were pointed out that a DApp must satisfy some requirements: (*i*) open source so that participants can audit the system, (*ii*) application operations and data are recorded and executed in a decentralised BC (e.g., using SCs), and (*iii*) a crypto

* Corresponding author.
E-mail addresses: n.truong@imperial.ac.uk (N. Truong), g.m.lee@ljmu.ac.uk (G.M. Lee), k.sun@imperial.ac.uk (K. Sun), f.guitton@imperial.ac.uk (F. Guitton), y.guo@imperial.ac.uk (Y. Guo).

[1] https://cointelegraph.com/news/report-ethereum-tron-and-eos-dominated-dapp-ecosystem-in-2019

[2] http://www.johnstonslaw.org

https://doi.org/10.1016/j.future.2021.05.025

token is used to access the service and to contribute to the operations (e.g., token reward) [7,8]. As of these features, ideally, DApps have the ability to operate without human intervention and to be self-sustaining because the participation of stakeholders is continuously strengthening the systems. According to *Vitalik Buterin*, DApps generally fall into two overlay categories, namely fully anonymous DApps and reputation-based ones [8]. The first category is DApps which participants are essentially anonymous and the whole service business logic is autonomously executed by a series of instant atomic operations. Pure financial services such as Bitcoin are examples of this. Another example is digital assets trading DApps such as software licence, data, and digitised properties in which the ownership can be impeccably transferred once a contract (defined and implemented using SCs) are performed [9].

The second category refers to a type of DApps which business logic requires a reputation-like mechanism to keep track of participants' activities for trust-related purposes. For instance, DApps for data storage and computation, similar to *Dropbox* and *Amazon AWS* in the centralised space, do require to maintain reputation-like statistic record of peers for service quality and security-related purposes (e.g., anti-DDoS). This requirement of trust is irrelevant to BC technology which supposedly ensures only data security (e.g., for distributed ledgers), autonomy and integrity of the business logic execution programmed in corresponding SCs. The quality of service (QoS) of such a DApp also depends on the service itself (i.e., how well the service handles the business logic defined in the SCs and caters to customers).

1.2. Necessity of a trust system in DApps ecosystem

DApps usage always comes with token movement from end-users to service contributors as a result of an incentive scheme, which is crucial to maintaining the service. However, due to the immutable nature, it is practically impossible to revoke any transaction once it is settled onto BC. Thus, a DApp has to make sure that end-users are dealing with trustworthy counter-parties before invoking any SCs' functions that can lead to a token payment. Intuitively, end-users tend to look for an indication of *'assurance'* before using any services. Indeed, a variety of DApps share the same stance on a challenge of lacking a unified decentralised framework to evaluate the trustworthiness of participants (for instance, decentralised storage and computing (similar to cloud storage like *Dropbox* and *Amazon AWS*), home-sharing (similar to *Airbnb*), car-sharing (similar to *Uber*), or a hotel distribution and reservation service (similar to *Booking.com*) backed by a BC platform). Therefore, a trust middleware that supports DApps' end-users to transact with trustworthy counterparts is of paramount importance as it penalises deceptive participants while encouraging authentic ones. In addition, a BC platform with a decentralised trust system will facilitate DApps to build up trust with their clients, which is fundamental in the business success.

1.3. Objectives and contributions

Our objectives are to envision and develop a universal decentralised system that operates along with any DApps to evaluate trust relationships between entities in the ecosystem. This trust system plays as middleware between a BC platform and DApps that provides mechanisms for DApps' end-users to build up and maintain a trust relationships network among the users. Operations of the system are fully decentralised, transparent, and accessible to all of the participants which are autonomously and flawlessly executed in a trustless fashion. It is also expected to effectively prevent from reputation attacks (e.g., Sybil, Whitewashing, Self-promoting, and Bad&Good-mouthing) and to dismiss masquerading hostile participants.

The main contributions of this paper are three-fold:

- Introduction to the concept and provision of a universal decentralised trust system that can be integrated into any DApps sharing a same Blockchain platform.
- A decentralised trust model with theoretical analysis, algorithms, and simulations.
- Providing the whole agenda of the real-world development of the system including technical solutions, implementation reference, as well as performance evaluation.

The rest of the paper is organised as follows. Section 2 briefly brings up background and related work and presents the provision and conceptual model of a decentralised trust system. Section 3 describes a system design with a trust evaluation model for the proposed system. Section 4 provides the algorithms and the theoretical analysis of the trust evaluation model. Section 5 is to discuss on the technical solutions and the implementation reference for the system development. Section 6 is dedicated to the system analysis and discussion. Section 7 concludes our work along with the future research directions.

2. Decentralised trust system provision for DApps ecosystem

To craft a BC platform into a mature DApp development environment, fundamental elements must be incorporated such as an Identity Management (IdM), a name registry, a wallet, a P2P messaging for end-users, a browser, and a decentralised trust/reputation system [8] (as illustrated in Fig. 1). These elements are core built-in services of a BC-based infrastructure for DApps development.

2.1. Related work

A large number of trust management mechanisms that have been proposed in various environments including social networks [10], P2P or ad-hoc networks [11], and IoT [12–14]. Those trust models could be adapted to different scenarios including BC-related environment. However, as the emerging BC technology is in the early stage, there is limited research on trust management for DApps. Most of the related research is to develop a trust or reputation management platform leveraging the advantages of BC such as decentralisation, immutability, trace-ability, and transparency. In this respect, researchers have proposed BC-based trust mechanisms to fortify specific applications in various environments including vehicular networks and intelligent transportation systems [15,16], wireless sensor networks [17,18], or IoT [19,20]. For instance, W. She et al. in [18] have proposed a BC-based trust model to detect malicious nodes in wireless sensor networks by implementing a voting mechanism on-chain, ensuring the trace-ability and immutability of voting information. M. Debe et al. have developed a reputation-based trust model built on top of Ethereum platform for fog nodes in a Fog-based architectural system [21]. The idea is similar in that a reputation mechanism, comprising of several SCs, is implemented on top of Ethereum platform so that clients can give feedback as ratings toward a Fog node when using a service provided by such node. The reputation of a fog node is simply accumulated on-chain from users' ratings. Being executed on-chain, such ratings and reputation values are immutably recorded in a decentralised fashion, thus ensuring data integrity as well as preventing from Denial of Service (DDoS) attack.

We, instead, look at a different angle of trust in BC-based applications in which a trust system plays a complementary component of the BC platform that cooperates with DApps to empower the ecosystem built on top of the platform. We target to develop a trust system for decentralised services in a BC ecosystem (e.g., Ethereum) in which participants (clients and service providers) interact with each other on-chain in a P2P manner.

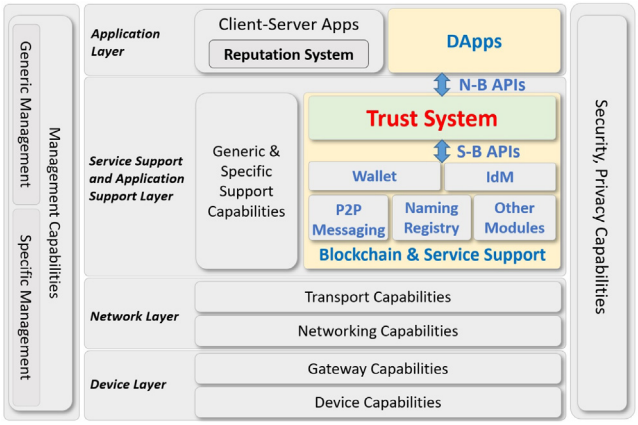

Fig. 1. Functional model of a BC-based infrastructure comprising of a trust system and other elements in alignment with IoT high-level architecture.

Our system plays as a unified trust solution working with any DApps. Our previous research in [9] has presented an introductory concept of a unified trust system to strengthen a BC platform. However, it has come without detailed analysis, algorithm, and technical solutions for the development of the decentralised trust system. In this paper, we further explore the concept and the feasibility of a unified trust system as middleware between a BC platform and DApps, as well as provide a proof-of-concept of the decentralised trust system along with the system design, algorithms, technical solutions and implementation reference.

2.2. High-level architecture of BC-based infrastructure and trust system

For a better understanding of the big picture of the whole BC-based infrastructure, we represent the high-level architecture of a full-stack IoT infrastructure by harmonising these components to the IoT and Smart Cities & Communities reference model.[3] As can be seen in Fig. 1, the BC infrastructure (equipped with some fundamental elements) is located in the *Service Support and Application Support*, which is a layer between the *Application* and *Network* layers in the IoT architecture, as one of the layer's component. DApps is located in the *Application* layer. Unlike client–server applications and services whose reputation/trust systems are separately developed, as depicted in Fig. 1, we envisage that DApps in the same ecosystem could leverage a universal trust system, which serves as a fundamental service for the BC-based infrastructure. This trust middleware exists because DApps' end-users in an ecosystem are identified by the same IdM and a name registry, and use the same cryptocurrency (e.g., provided by a BC platform) to consume the services.

2.3. High-level architecture of trust system

In this sub-section, fundamental elements of a decentralised trust middleware between a BC platform and DApps are described. As can be seen in Fig. 2, the proposed system consists of two basic components named *Data Collection & Extraction* and *Trust Evaluation* that collect and aggregate necessary trust-related information and evaluate trust relationships, respectively. These two components are along with *North-bound* and *South-bound* APIs for providing trust-related services to DApps and for collecting data from a BC or applications and services, respectively.

[3] http://itu.int/en/ITU-T/studygroups/2017-2020/20/Pages/default.aspx

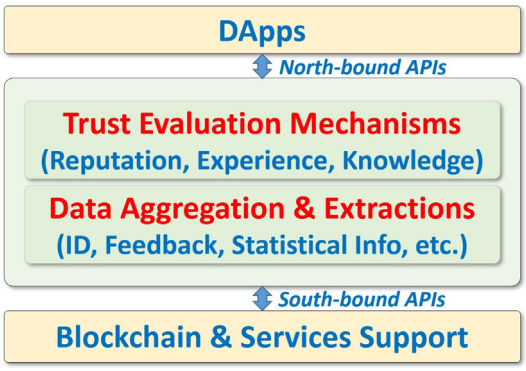

Fig. 2. Conceptual model of the proposed trust system.

2.3.1. Trust evaluation mechanism

We adopt the *REK* trust model proposed in [13,14] to the DApps ecosystem scenario in which both *trustors* and *trustees* are end-users of DApps. In the REK model, a trust relationship is evaluated by assembling three indicators called Reputation (of the trustee), Experience and Knowledge (of the trustor toward the trustee). In DApps scenarios, there is limited availability (or difficult to obtain) of off-chain information (i.e., information that is recorded outside BC) for evaluating Knowledge indicator because users' identity is pseudo-anonymised and challenging to link to outside world [22]. Instead, transactions between end-users are immutably recorded and publicly available on-chain, which can be leveraged for Experience and Reputation evaluations. Therefore, we employ an adoption of the *REK* trust evaluation model called *DER* which only utilises two indicators Experience and Reputation in decentralised environment. Details of the *DER* trust system is described in the next section.

Generally, after each transaction between two entities in a DApp, the trust system enables an entity to give feedback toward its counterpart, thus establishing and updating the *Experience* relationship between the two. As a result, the trust system maintains an *Experience* network among participants, which is publicly recorded on-chain and autonomously updated whenever an entity gives feedback to the other. *Reputations* of all participants are then calculated based on the Experience network, following the idea of weighted Google PageRank algorithms [23,24]. Finally, trust relationship between two entity is calculated as a composition between *Experience* and *Reputation*.

2.3.2. Data collection and extraction

By nature, a BC is a record of a continuous growing list of transactions among end-users which can be analysed to extract a network topology of end-user interactions. Nonetheless, further information about QoS is required to be collected and aggregated in order for the *DER* trust evaluation mechanism to be performed. Therefore, a decentralised feedback mechanism associated with DApps in a BC platform is required to reflect QoS once end-users (e.g., service clients) successfully carry out transactions with their counterparts (e.g., DApp providers). This mechanism creates a *distributed ledger* that logs users' feedback (toward a DApps service) along with the information about associated transactions (e.g., end-user ID (*from* address), counterpart ID (*to* address), and *timestamp*). Feedback can be either implicit or explicit which may or may not require human participation [25]. The trust system then extracts feedback and transactions information recorded in BC as the inputs for the *DER* trust evaluation model (i.e., the Experience and Reputation calculations) in order to evaluate trust relationships between any two participants in the decentralised ecosystem.

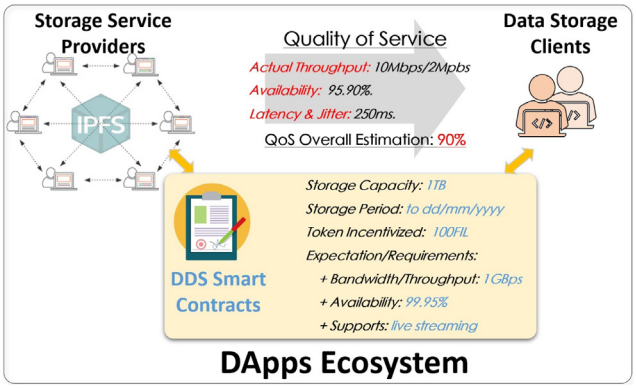

Fig. 3. Case study of a decentralised storage service built on top of a BC platform that incentivises storage nodes with crypto-tokens. A decentralised trust system should be incorporated in order for clients to select trustworthy providers whiles penalising corrupt ones.

3. System design and DER trust model

This section introduces the high-level system design of the proposed trust system along with an evaluation model.

3.1. Case study

For a better interpretation, we scrutinise the decentralised data storage services (DDS), in regard to some projects being developed and implemented in the real-world like Storj,[4] Sia,[5] and Filecoin[6] (built on top of the InterPlanetary File System[7] (IPFS)). Decentralised storage is a promising solution to cooperate or even to take over the conventional centralised cloud storage where data is split into multiple chunks and distributed to storage nodes across a P2P network. These storage nodes, as DDS providers, are expected to reliably store the data as well as provided reasonable network bandwidth with appropriate responsiveness for data owners to retrieve their data. As a reward, such storage nodes are incentivised by crypto-tokens. It is worth noting that end-users in DApps ecosystem can be both data owners (DDS clients) and storage nodes (DDS providers). The decentralised storage concept is similar to the legacy P2P file sharing such as BitTorrent[8] but fortified with advanced cryptography and encryption mechanisms as well as incentive schemes built upon a BC platform. It is expected to solve the long-standing challenges of single-point-of-control and -failure in centralised data silos, and to bring essential control of data back to the owners.

As illustrated in Fig. 3, the DDS deploys necessary SCs on top of a BC platform to execute the business agreement between DDS clients (i.e., data owners) and DDS providers (i.e., storage nodes) such as *storage space and period, guaranteed performance (e.g., availability, throughput, bandwidth, and latency)*, and the *Incentive scheme* (i.e., *Token Reward*). Unfortunately, such SCs are *unable to ensure* the QoS of the DDS service provided by a set of storage nodes because *(i)* it is impractical for the SCs to monitor and enforce the performance of the DDS providers, and *(ii)* the guaranteed performance can only be measured once the SCs are already invoked. In this regard, a trust system that manages the performance history of the storage nodes and ranks them in order of trustworthiness (to provide high QoS) is of paramount importance.

4 https://storj.io
5 https://sia.tech
6 https://filecoin.io
7 https://ipfs.io
8 https://en.wikipedia.org/wiki/BitTorrent

Table 1
Notations used in the experience model.

Notation	Description
Exp_t	Experience value at time t, Exp_0 is the initial value
min_{Exp}	minimum Exp value, $min_{Exp} = 0$ if Exp is normalised in [0,1]
max_{Exp}	maximum Exp value, $max_{Exp} = 1$ if Exp is normalised in [0,1]
ϑ_t	Feedback score at time t
α	Maximum increase value of Exp in two consecutive transactions, $0 < \alpha < max_{Exp}$
β	Decrease rate, $\beta > 1$
θ_{co}	Cooperative threshold for a feedback score ϑ_t. A feedback is cooperative if $\vartheta_t \geq \theta_{co}$
θ_{unco}	Uncooperative threshold for a feedback score ϑ_t. A feedback is uncooperative if $\vartheta_t \leq \theta_{unco}$
δ	Minimum Decay value ensuring any Experience relationship degenerates if it is not maintained
γ	Decay rate controlling the amount of the decay

3.2. DER trust model

In the DER model, trust relationship between two entities is a compound of two elements: *Experience* (of a trustor toward a trustee) and *Reputation* (of a trustee). This section describes mechanisms to calculate such two elements.

3.2.1. Experience mechanism

Experience is an asymmetric relationship from an entity to the another which is built up from previous transactions between the two. Experience is an indicator of trust [13]. For instance, an experience (denoted as $Exp(A, B)$) is constituted from a DDS client (i.e., a data owner, denoted as A) to a DDS provider (i.e., a storage node, denoted as B) once A invokes an SC to use the storage service offered by B. Higher $Exp(A, B)$ value represents higher degree of trust from A to B. Essentially, $Exp(A, B)$ increases if B provides high-quality storage service to A (which is reflected by a feedback score ϑ_t) and vice versa. It is worth noting that feedback can be provided by either clients (e.g., A) or an authorised third-party who is monitoring performance of service providers (e.g., B). Also, $Exp(A, B)$ gets decay if no transactions taken place after a period of time or a transaction is neutral (i.e., neither cooperative nor uncooperative). The amount of increase, decrease and decay depends on intensity of transactions, feedback scores ϑ, and the current value of $Exp(A, B)$ which can be modelled by linear difference equations and a decay function as follows (notations are denoted in Table 1) [13,14]:

- **Increase model**

The current $Exp(A, B)$ (denoted as Exp_{t-1}) increases when there occurs a cooperative transaction (at the time t, indicated by the feedback score $\vartheta_t \geq \theta_{co}$) that follows the linear difference equation:

$$Exp_t = Exp_{t-1} + \vartheta_t \Delta Exp_t \tag{1}$$

where ΔExp_t is defined as follows:

$$\Delta Exp_t = \alpha(1 - \frac{Exp_{t-1}}{max_{Exp}}) \tag{2}$$

- **Decrease model**

Similarly, $Exp(A, B)$ decreases if the transaction is uncooperative (indicated by the feedback score $\vartheta_t \leq \theta_{unco}$), following the equation:

$$Exp_t = Max(min_{Exp}, Exp_{t-1} - \beta(1 - \vartheta_t)\Delta Exp_t) \tag{3}$$

in which ΔExp_t is specified in Eq. (2). The decrease rate $\beta > 1$ implies that it is easier to lose the $Exp(A, B)$ value due to an uncooperative transaction than to gain it (by a cooperative transaction).

- **Decay model**

$Exp(A, B)$ decays if there is no transaction after a period of time or a feedback is neutral (i.e., $\theta_{unco} < \vartheta < \theta_{co}$) and the decay rate is assumed to be inversely proportional to the strength of the experience relationship (i.e., Exp_t value) [26]. Based on these observations, the Decay model is proposed as follows:

$$Exp_t = Max(min_{Exp}, Exp_{t-1} - \Delta Decay_t) \tag{4}$$

$$\Delta Decay_t = \delta(1 + \gamma - \frac{Exp_{t-2}}{max_{Exp}}) \tag{5}$$

3.2.2. Reputation mechanism

The reputation of an entity represents the overall perception of a community regarding the characteristic of the entity such as trustworthiness. In the DApps ecosystem, the reputation of an end-user U (denoted as $Rep(U)$) can be calculated by aggregating $Exp(i, U)$, $\forall i$ are users who have already been transacted with U. To calculate the reputation of end-users, we utilise the model proposed in [13,14] which is based on the standard PageRank [24] and the weighted PageRank [23,27].

Let N be the number of end-users in the DApps ecosystem, an directed graph $G(V, E)$ is constructed in which V is a set of N users, $E \subseteq \{(x, y)|(x, y) \in V^2 \wedge x \neq y\}$ is set of edges representing experience relationship $E(x, y) = Exp(x, y)$. If there is no prior transaction between (x, y); $E(x, y) = 0$. To enable the reputation model, $G(V, E)$ is divided into two sub-graphs: positive experience $PG(V, PE)$ in which any edge $PE(x, y) = Exp(x, y)$ satisfying $Exp(x, y) > \theta$ and negative experience $NG(V, NE)$ in which any edge $NE(x, y) = Exp(x, y)$ satisfying $Exp(x, y) < \theta$, where θ is a predefined threshold. d parameter is a damping factor $(0 < d < 1)$ introduced in standard PageRank [24]. The reputation for each sub-graph is then calculated as follows:

- **Positive Reputation**

$$Rep_{Pos}(U) = \frac{1-d}{N} + d(\sum_{\forall i} Rep_{Pos}(i) \times \frac{PE(i, U)}{C_{Pos}(i)}) \tag{6}$$

in which $C_{Pos}(i) = \sum_{\forall j} PE(i, j)$ representing the sum of all positive experience values that the end-user i holds (toward other end-users).

- **Negative Reputation**

$$Rep_{Neg}(U) = \frac{1-d}{N} + d(\sum_{\forall i} Rep_{Neg}(i) \times \frac{1 - NE(i, U)}{C_{Neg}(i)}) \tag{7}$$

in which $C_{Neg}(i) = \sum_{\forall j}(1 - NE(i, j))$ representing the sum of all complements of negative experience values (i.e., $1 - NE(i, j)$) that the end-user i holds (toward other end-users).

- **Overall Reputation**

$Rep(U)$ is the aggregation of $Rep_{Pos}(U)$ and $Rep_{Neg}(U)$:

$$Rep(U) = max(0, Rep_{Pos}(U) - Rep_{Neg}(U)) \tag{8}$$

3.2.3. Trust aggregation

Trust relationship between trustor A and trustee B is a composite of $Exp(A, B)$ and $Rep(B)$:

$$Trust(A, B) = w_1 Rep(B) + w_2 Exp(A, B) \tag{9}$$

in which w_1 and w_2 are weighting factors satisfying $w_1 + w_2 = 1$. It is worth noting that any end-user once signing up for a DApp is assigned a default value at bootstrap (e.g., $\frac{1}{N}$). If A and B have no prior transaction then $Exp(A, B) = 0$. In this case, $w_1 = 1$ and $w_2 = 0$; thus, $Trust(A, B) = Rep(B)$.

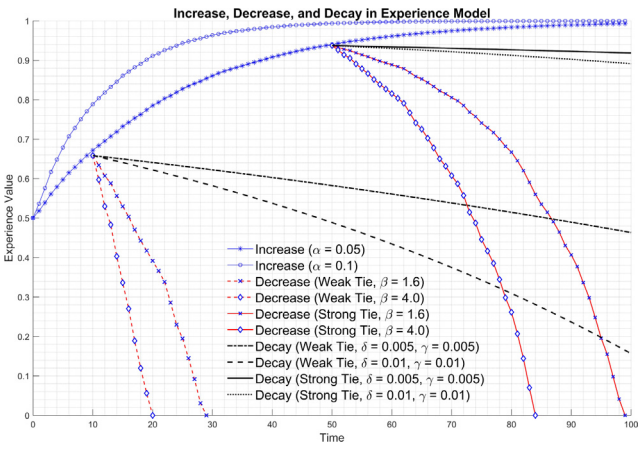

Fig. 4. Increase, Decrease, and Decay in Experience relationship.

4. Trust model: Evaluation and simulation

This section provides detailed evaluation of the *DER* trust model including model equations analysis, algorithms, and simulation of the Experience and Reputation models.

4.1. Experience model

4.1.1. Analysis

For simplicity, Exp values and feedback score ϑ are normalised to the range (0, 1) with $max_{Exp} = 1$, $min_{Exp} = 0$ and the initial value $0 < Exp_0 < 1$. We then have:

Lemma 4.1. *The Increase model (defined in Eq. (1)) is (*) a monotonically increasing function and (**) asymptotic to* 1.

The proof of this lemma is provided in Appendix A.1. As the Increase model is monotonically increasing, it is obvious that the Decrease model defined in Eq. (3), which is based on ΔExp_t in Eq. (2), is decreasing. The decrements depend on the current Exp_t value and the uncooperative ϑ_t feedback score. The decrease rate β depicts the ratio of the decrements compared to the increments, which is normally greater than 1 as the current experience Exp_t is *"difficult to gain but easy to loose"*.

The Decay model defined in Eq. (4) ensures that an experience relationship gets weakened if there is no or neutral transactions after a period of time. This is because the decay value $\Delta Decay_t$ specified in Eq. (5) is always > 0 as $0 < Exp_{t-2} < 1$ $\forall t \geq 2$; and it is inversely proportional to Exp_{t-2}, implying that a strong relationship persists longer than a weak one.

4.1.2. Algorithm and simulation

Based on the Experience model defined in Section 3.2.1 along with the analysis, the algorithm calculates experience value $Exp(A, B)$ of entity A toward entity B is demonstrated in mathematical-style pseudo-code as in Algorithm 1. It is worth noting that the parameters controlling the Experience model are preset for our demonstration and should be optimised for specific scenarios.

For demonstration purposes, the algorithm is implemented in *Matlab* with different controlling parameters settings. As depicted in Fig. 4, two sets of parameters configuration are taken into account in which the maximum increase value α is either 0.05 or 0.1, the decrease rate β is either 1.6 or 4.0, and the parameter pair for the decay model (δ, γ) is either (0.005, 0.005) or (0.01, 0.01). The initial value is preset $Exp_0 = 0.5$. As can be seen in Fig. 4, the results show that both increase model curves are asymptotic

Alg. 1: Experience Calculation Algorithm

Input : Current experience value Exp_{t-1}
Previous experience value Exp_{t-2}
Feedback score ϑ_t
Output: Updated experience value Exp_t

Parameters Preset
$Exp_0 = 0.5$; ▷ In case there is no prior transaction, Exp_{t-1} and Exp_{t-1} are set to Exp_0;
$min_{Exp} = 0$; $max_{Exp} = 1$; ▷ Experience value is normalised in the range [0,1];
$\theta_{co} = 0.7$; $\theta_{unco} = 0.5$;
$\alpha = 0.05$; $\beta = 1.6$;
$\delta = 0.005$; $\gamma = 0.005$
Begin
if $\vartheta_t \geq \theta_{co}$ **then**
    ▷ Increase Model;
    $Exp_t = Exp_{t-1} + \vartheta_t \alpha (1 - \frac{Exp_{t-1}}{max_{Exp}})$
else if $0 < \vartheta_t \leq \theta_{unco}$ **then**
    ▷ Decrease Model;
    $Exp_t = Max(min_{Exp}, Exp_{t-1} - \beta(1 - \vartheta_t)\alpha(1 - \frac{Exp_{t-1}}{max_{Exp}})$
else
    ▷ No transaction ($\vartheta_t = 0$) or neutral $\theta_{unco} < \vartheta_t < \theta_{co}$
    ▷ Decay Model;
    $Exp_t = Max(Exp_0, Exp_{t-1} - \delta(1 + \gamma - \frac{Exp_{t-2}}{max_{Exp}})$
Return Exp_t

to 1, which is already proven in the theoretical analysis, at different rates depending on the controlling parameter α. The results also indicate that stronger experience relationships require more cooperative transactions to achieve. For instance, with $\alpha = 0.05$, experience value increases from 0.5 to 0.7 after 12 consecutive transactions whereas it increases from 0.9 to just 0.94 after the same number of transactions.

The simulation results of the Decrease model show that experience relationships are prone to uncooperative transactions suggesting that a strong tie is hard to attain but easy to lose, particularly with higher decrease rate β. For instance, with $\alpha = 0.05$ and $\beta = 4.0$, it takes 50 consecutive cooperative transaction to increase the experience value from 0.5 to 0.9 but takes only 22 uncooperative transactions to drop from 0.9 to 0.5. As can also be seen from the figure, both decrease and decay models exhibit a same behaviour that a strong tie is more resistant to uncooperative transactions/decay whereas a weaker one is more susceptible. These characteristics of the experience model manifest the human social relationships, showing the practicability of the proposed model.

4.2. Reputation model

4.2.1. Analysis

Denote $(N \times 1)$ column vectors *Rep*, Rep_{Pos}, and Rep_{Neg} whose elements are overall reputation, positive reputation, and negative reputation of N end-users in DApp ecosystem, respectively. As specified in Eq. (6), $Rep_{Pos}(U)$ of the user U is calculated from others' positive reputations $Rep_{Pos}(i)$ $\forall i$ holding positive experience $PE(i, U)$ with U. Consequently, there would be correlations among the N positive reputations, which would lead to the fact that Rep_{Pos} might not exist or might be ambiguous (i.e., there exists more than one values for a user that satisfy Eq. (6)). The same condition could happen for Rep_{Neg}, and for *Rep*, as a consequence. We then need to prove this following lemma:

Lemma 4.2. *The reputation vector Rep exists and is unique.*

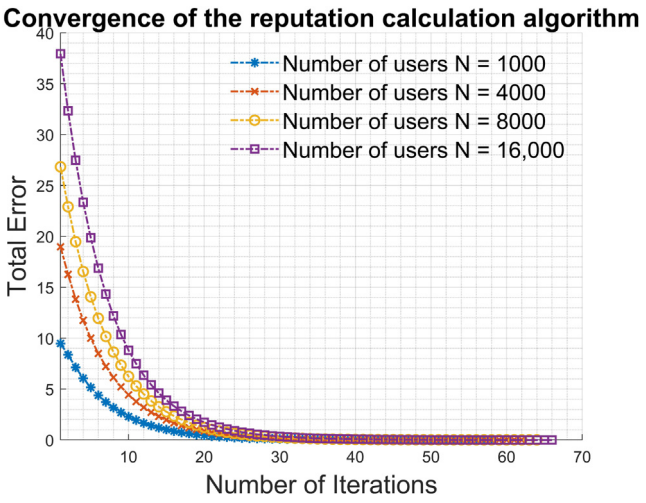

Fig. 5. Convergences of the reputation algorithm using interactive method with different sizes of DApp ecosystem.

The proof of this lemma is provided in Appendix A.2. As the existence and the uniqueness are proven, the reputation vector *Rep* of N end-users in DApps ecosystem can be calculated by solving the matrix equations defined in Eqs. (6), (7). The traditional algebra method to solve an *NxN* matrix equation (e.g., Eq. (6) or Eq. (7)), whose the complexity is $\mathcal{O}(N^3)$, is impractical when the size of the DApp ecosystem is enormous (e.g., in millions). Instead, the reputations of the N end-users can be approximately calculated with a predefined accuracy tolerance using an iterative method, which is much more efficient [28,29]. The latter approach is utilised in our system to solve Eqs. (6) and (7), depicted by Algorithm 2 in the next subsection.

4.2.2. Algorithm and simulation

The reputation calculation algorithm (Algorithm 2) takes the current positive and negative Reputation values defined in Eq. (6) and (7) (i.e., Rep_{Pos} and Rep_{Neg} $N \times 1$ column vectors) with the Experience network (represented by $(N \times N)$ matrix E) as inputs and outputs the updated Rep_{Pos} and Rep_{Neg}. Detailed explanation of the algorithm is commented out in the pseudo-code. Finally, as defined in Eq. (8), the overall reputation for N end-users (i.e., $N \times 1$ column vector *Rep*) is then simply obtained by adding two vectors Rep_{Pos} and Rep_{Neg}, which are the outputs of Algorithm 2.

The simulation of the proposed reputation calculation algorithm are conducted for different DApp ecosystem sizes (i.e., $N =$ 1000, 4000, 8000 and 16,000) with the *error tolerance tol* $= 10^{-5}$, which is accurate enough to rank N end-users in the DApp ecosystem. As depicted in Algorithm 2, the total error *err* is calculated as the Euclidean norm of the vector difference of the *Rep* vector in two consecutive iterations. Fig. 5 illustrates the convergence rate of the algorithm, showing the rapid reduction of the total error as more iterations are carried out. As can be seen from the figure, the algorithm converges in less than 70 iterations (to be exact: 54, 61, 64, and 66 iterations) for four DApps ecosystem sizes $N =$ 1000, 4000, 8000 and 16,000, respectively. These results suggests that the reputation model well scales for a huge network as the scaling factor is roughly linear in *logN*.

5. Technical solutions and implementation

This section provides a real-world demonstration for the proposed decentralised trust system and how a decentralised storage service based on *IPFS* interacts with it. The demonstration is

Alg. 2: Reputation algorithm using iterative method

Input : $(N \times N)$ matrix E (set of edges in the directed graph $G(V, E)$ of N end-users)
Positive reputation $N \times 1$ vector Rep_{Pos}
Negative reputation $N \times 1$ vector Rep_{Neg}
Output: Updated Rep_{Pos} and Rep_{Neg}

Parameters Preset
$d = 0.85$; ▷ damping factor in standard PageRank
$tol = 1e-5$; ▷ Error tolerance
$thres = 0.5$; ▷ threshold for positive and negative experience
Begin
▷ Elicit matrices PE and NE from matrix E;
$PE = zeros(N, N)$; ▷ initialise zero matrix for NE
$PE = zeros(N, N)$; ▷ initialise zero matrix for PE
for $i \leftarrow 1$ *to* N **do**
  for $j \leftarrow 1$ *to* N **do**
    if $E(i, j) \geq thres$) **then**
      $PE(i, j) = E(i, j)$
    else if $0 < E(i, j) < thres$ **then**
      $NE(i, j) = 1 - E(i, j)$
▷ Constitute $1 \times N$ row vectors C_{Pos} and C_{Neg};
$C_{Pos} = zeros(1, N)$; ▷ initialise zero vector for C_{Pos}
$C_{Neg} = zeros(1, N)$; ▷ initialise zero vector for C_{Neg}
for $i \leftarrow 1$ *to* N **do**
  for $j \leftarrow 1$ *to* N **do**
    $C_{Pos}(1, i) = C_{Pos}(1, i) + PE(i, j)$;
    $C_{Neg}(1, i) = C_{Neg}(1, i) + NE(i, j)$;
▷ Constitute transition matrices of PE and NE;
for $i \leftarrow 1$ *to* N **do**
  for $j \leftarrow 1$ *to* N **do**
    if $PE(j, i) > 0$) **then**
      $A_{Pos}(i, j) = \frac{PE(j,i)}{C_{Pos}(1,j)}$; ▷ Transition matrix for PE
    if $NE(j, i) > 0$) **then**
      $A_{Neg}(i, j) = \frac{NE(j,i)}{C_{Neg}(1,j)}$; ▷ Transition matrix for NE
▷ Update Rep_{Pos} and Rep_{Neg} based on Eqs. (6) and (7);
$I = ones(N, 1)$; ▷ create vector of all ones
$err = 1$; ▷ Total error of the current iteration
while $err \geq tol$ **do**
  $temp_{Pos} = d \times A_{Pos} \times Rep_{Pos} + \frac{(1-d)}{N} \times I$;
  $temp_{Neg} = d \times A_{Neg} \times Rep_{Neg} + \frac{(1-d)}{N} \times I$;
  ▷ update err using $\mathcal{N}(v)$ function as the Euclidean norm of vector v;
  $err = \mathcal{N}(temp_{Pos} - Rep_{Pos}) + \mathcal{N}(temp_{Neg} - Rep_{Neg})$;
  $Rep_{Pos} = temp_{Pos}$; ▷ update Rep_{Pos} vector
  $Rep_{Neg} = temp_{Neg}$; ▷ update Rep_{Neg} vector
Return $[Rep_{Pos}, Rep_{Neg}]$

carried out on top of the Ethereum permissionless BC platform in which system components, functionality, technical challenges and solutions are identified as the implementation reference for developers who wish to build a similar system. Source-code of the demonstration can be found at Github repo.[9]

5.1. System setup

The DDS service and the proposed decentralised trust system are implemented on top of the permissionless Ethereum platform to which fundamental elements for developing a DApp have already been deployed. For instance, in our platform setup,

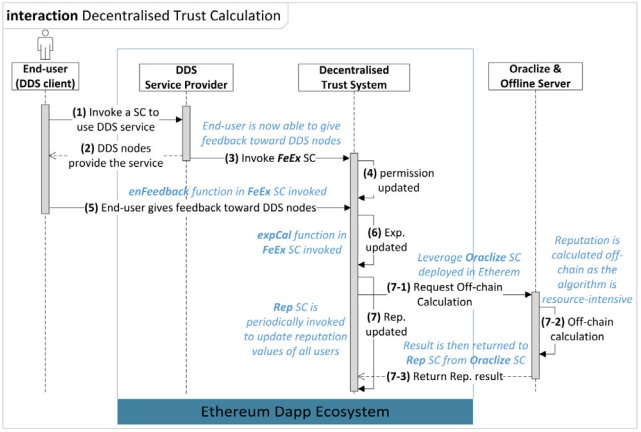

Fig. 6. Sequence diagram of how the decentralised trust system is incorporated with the DDS service and how the proposed *DER* trust calculation is performed.

Ethereum *account* and *address* are leveraged for IdM, *Metamask*[10] is for BC browser and a wallet service, and *web3*/*web3j*[11] are DApps APIs for interacting with Ethereum network (e.g., SCs and end-users). SCs are implemented in Solidity using Truffle suite framework[12] and deployed in an Ethereum test-net (i.e., we use several test-nets including *Ropsten*, *Kovan Rinkeby*, and *Goerli*) for real-world experience. We assume that IPFS storage nodes are also clients of the DApps ecosystem (e.g., Ethereum clients in *Ropsten*, *Kovan* or *Rinkeby* test-net) that get incentivised once providing storage capability (e.g., IPFS storage nodes *host* and *pin* the hash of requested files from data owners).

The overall procedure of the setting system is illustrated in Fig. 6. As can be seen in the sequence diagram, a client starts to use the DDS service by making a transaction to a DDS SC (step **(1)**), which invokes **enFeedback** function in ***FeEx*** SC of the trust system to grant the client permission to give feedback to the DDS nodes ((step **(3)**), **(4)**)). Once getting feedback from the end-user (step **(5)**), experience relationships between the user and the DDS nodes are updated on-chain by executing **expCal** function in ***FeEx*** SC (step **(6)**). On the contrary, as the reputation calculation is resource-intensive, it is impractical to implement the algorithm (i.e., Algorithm 2) on-chain; instead, only the results (i.e., reputation values of entities) are publicly recorded on-chain. This challenge can be circumvented by using Oraclize service, as demonstrated in step **(7-1)**, **(7-2)**, and **(7-3)** in Fig. 6. With the same reason, **Rep** SC is not invoked whenever an experience relationship is updated; instead, it is periodically self-executed — for example, for every 100 blocks.

5.2. Feedback and experience smart contract

This SC, denoted as *FeEx*, contains feedback information and experience relationship of any entity A (i.e., a DDS client) toward entity B (an IPFS storage node) where a transaction between A and B has been carried out (i.e., A uses the DDS service provided by B depicted by step **(1)** and **(2)** in Fig. 6). *FeEx* SC also provides functions for end-users to give feedback and to update experience relationships accordingly (step **(3)** to **(6)** in the sequence diagram). Note that A and B are identified by Ethereum *address* in the ecosystem.

9 https://github.com/nguyentb/Decentralised_Trust_Eth_IPFS.git

10 https://metamask.io/
11 https://github.com/web3j/web3j
12 https://truffleframework.com

5.2.1. Ledger data model

Necessary information about users' feedback and experience relationships is permanently recorded on-chain using state variables defined in *FeEx* SCs. These state variables are as a public *distributed ledger* comprised of the full history of *state transitions* of all experience relationships between any two entities. It is convenience to obtain the latest information of any experience relationship as Ethereum supports *key–value* data format and the latest state of the ledger (recording the most recent experience relationships information) can be found in the most recent block.

FeEx SC stores a state variable called *FeExInfo* in its contract storage in form of nested key–value pairs using Ethereum built-in *mapping* type as in Appendix B.1. *FeExInfo* consists of information about the relationship from A toward B, specified in *FeExStrut* data structure: *(ii)* $Exp(A, B)$ value, *(iii)* feedback score, and *(iv)* a flag indicating whether A has permission to give B feedback. Any parties or SCs can easily access *FeExInfo* recorded on-chain to obtain desired information for their purposes.

5.2.2. Functionality

The *FeEx* SC contains two main functions: *(i) enFeedback* enables/revokes permission of a data owner A to give feedback to storage node B by updating the permission flag in *FeExInfo* with associated transaction ID; and *(ii) expCal* calculates $Exp(A, B)$ value and updates *FeExInfo* whenever A gives feedback to B. The *enFeedback* function is called by an SC of the DDS service once a transaction has been carried out (illustrated by step **(3)** in Fig. 6).

The *expCal* implements the experience calculation function following Algorithm 1 proposed in Section 4.1. It is worth noting that as there is no global time server synchronised among nodes in the Ethereum BC platform so that the implementation of the decay model is not straightforward. To circumvent this challenge, *expCal* determines *time* in Algorithm 1 using block height (*block.number* property) so that $Exp(A, B)$ decays every a number of blocks if no transaction occurred between A and B during the period.

5.3. Reputation smart contract

Reputation SC, denoted as *Rep*, records positive reputation and negative reputation of all users (e.g., IPFS storage nodes) using two state variables *RepPosInfo* and *RepNegInfo*, respectively. *Rep* SC is invoked in step **(7)** which consists of three subsequent steps, namely **7-1**, **7-2**, and **7-3** as illustrated in the sequence diagram (Fig. 6).

5.3.1. Ledger data model

The data model for the two state variables *RepPosInfo* and *RepNegInfo* is a mapping between a user' address and a value, depicted in Appendix B.2. These two state variables play the role of a public *distributed ledger* permanently recording a full history of *state transitions* of the positive and negative reputation of all users.

5.3.2. Functionality

The reputation calculation algorithm (Algorithm 2) performs matrix multiplication with numerous iterations that requires a large number of operations and local variable manipulations. Consequently, the resource-consumption and the *gas* cost for executing this algorithm on-chain are extremely high, which is infeasible to be implemented in *Rep* SC. To bypass this challenge, off-chain storage and calculations appear as a promising solution. The catalyst of this solution is that high-volume data and resource-intensive tasks should be stored and processed off-chain; only results of the off-chain tasks are piggybacked for on-chain ledgers and/or calculations. However, as an SC must be *deterministically* executed, there might be a room for ambiguity if SC executions rely on information from off-chain sources. In addition, this practice could turn a decentralised system into a centralised one due to the dependency on an external source of information. This dilemma is known under the term: *"Oracle problem"* [30]. The following section will describe in detail how *Rep* SC can accomplish the off-chain reputation calculation while mitigating the Oracle problem.

5.4. Off-chain computation for reputation

Oracle problem could be mitigated by leveraging a decentralised trusted provider to feed required data into SCs. For instance, Oraclize.[13] deploys an SC on Ethereum platform as an API for other SCs to interact with the *outside world*[14] The Oraclize SC works as a bearer that gets required data from an external source and delivers the data to the requested SCs in a decentralised fashion. Furthermore, to alleviate the ambiguity, it *(ii)* provides *authenticity proof* as an assurance for data integrity. In the implementation, we follow this Oraclize solution to calculate users' reputations off-chain.

Assume that there is already an off-chain server, called *RepCalService*, that implements Algorithm 2 to calculate positive and negative reputations and provides an API (e.g., REST API) to retrieve the calculation results. The implementation of the off-chain service is straightforward: it queries the Ethereum BC to obtain experience relationships stored in *FeExInfo* and the current reputations values from *RepPosInfo* and *RepNegInfo* state variables as inputs for Algorithm 2. *Rep* SC then periodically calls this service to update the reputation values in a decentralised fashion using Oraclize solution. The implementation reference depicted in Appendix B.3 shows how to execute these tasks. Specifically, *Rep* interacts with the Oraclize service by importing the Oraclize SC (i.e., *provableAPI.sol*) to make a query to *RepCalService* using *oraclizeQuery()* function. A *callback* function also needs to be implemented in order to get the results from the query and to update *RepPosInfo* and *RepNegInfo* accordingly.

5.5. Integration of DDS service and trust system

Supposedly, the DDS service implements some SCs for data storage business logic between data owners and storage nodes, which is out of the scope of this paper. The main focus of the paper is that once a transaction has been accomplished between a client and an IPFS storage node, the *enFeedback* function in the *FeEx* is invoked that enables the owner to give feedback to its counterpart, which will establish experience and trust relationships (step **(2)** in Fig. 6). For this reason, a DDS SC (i.e., the caller SC) defines an interface of *FeEx* SC (i.e., the callee SC) and calls it with the callee's contract address as demonstrated in Appendix B.4.

Similarly, when a data owner gives feedback toward a storage node (with value *fbScore*), DDS invokes *expCal* function that calculates the experience relationship between the two and updates *FeExInfo* accordingly. In the demonstration, feedback scores are randomly generated; however, in the real-world scenarios, a function to measure DDS QoS shall be implemented to correctly reflect the service quality. As Solidity supports interactions between SCs deployed on Ethereum platform, the proposed trust system is feasibly actualised as any DApps including DDS can be incorporated by invoking public functions or accessing trust-related information from state variables defined in the SCs of the proposed trust system.

[13] https://docs.provable.xyz/

[14] https://github.com/provable-things/ethereum-api/blob/master/oraclizeAPI_0.4.sol

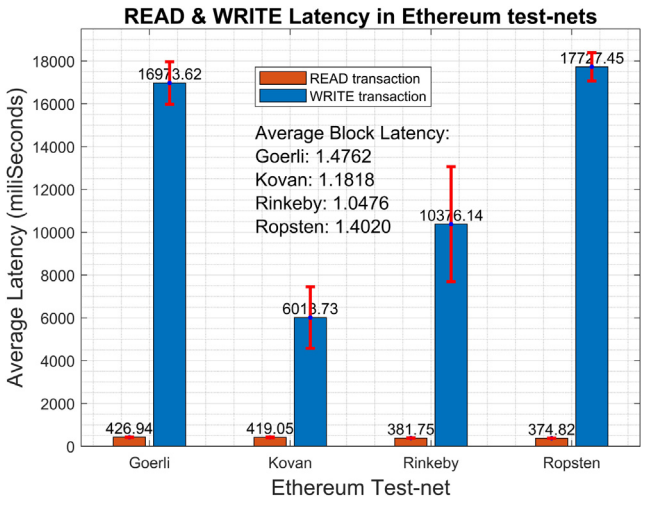

Fig. 7. Latency of READ and WRITE from/to Smart Contracts in Ethereum test-nets.

Finally, to reinforce service quality for a client, the DDS service queries *RepPosInfo*, *RepNegInfo* and *FeExInfo* stored at *FeEx* and *Rep* SCs, respectively, to receive reputation and experience values related to this client. The DDS service then aggregates this information for finalising trust values between the client and the storage nodes and provides the most trustworthy counterparts to the client.

6. System analysis and discussion

The demonstration system presented in Section 5 is a proof-of-concept of a universal decentralised trust system which is incorporated into a BC infrastructure as an underlying service for supporting DApps. This section investigates and discusses on the practicality, performance, and security-related aspects of the proposed trust system.

6.1. Performance evaluation, feasibility and limitation

In order to illustrate the real-world performance, we deploy our system to different BC platforms, i.e., Ethereum test-nets namely *Ropsten*, *Kovan*, *Rinkeby*, and *Goerli*. We carry out latency measurement of both *READ* and *WRITE* transactions to the ledger *FeExInfo* in the *FeEx* SC in the four test-nets. The results are shown in Fig. 7. The performance measurement script can also be found at the same repo.[15]

It is worth noting that in *READ* transactions, an Ethereum platform does not perform the consensus mechanism; instead, in *WRITE* transactions, consensus mechanism (i.e., Proof-of-Work (Ethash) in *Ropsten*, Proof of Authority (Authority Round) in *Kovan*, Proof of Authority (Clique) in both *Rinkeby*, and *Goerli*) is carried out as the state of the ledger is changed. In details, *WRITE* transactions require further complicated processes including block formulation and mining, broadcast the mined block to peers in the network, block verification, and updating the ledger. This is why the latency of *READ* transactions is much smaller than *WRITE* transactions, reassured by the results in Fig. 7. As can be seen in the figure, the average latency of *READ* transactions is roughly the same in all four test-nets at around *350–420* ms with relatively small standard deviations. This indicates the consistency when querying data from the ledger. Compared to *READ* transactions, the average latency in *WRITE* transactions is significantly risen to 6013, 10376, 16973, and 17727 ms, which is 15 to 42 times higher, in *Kovan*, *Rinkeby*, *Goerli*, and *Ropsten*, respectively. The standard deviations, however, are different in the four test-nets: *Ropsten* and *Goerli* introduce considerably higher *WRITE* latency compared to Kovan and Rinkeby (2–3 times) but *WRITE* transactions are more stable as the standard deviations are small. Particularly, in *Rinkeby* test-net, the standard deviation is substantially high — The latency spreads out in a wide range, from *4500* to *17350* ms.

Results also show the *block latency*[16] in *WRITE* transactions in the four test-nets. In Kovan and Rinkeby, *WRITE* transactions are almost appended and confirmed in the next block demonstrated by block latency is close to 1 whereas in Goerli and Ropsten, it could take one or two more blocks before the transaction is written onto a new block. This is probably one of the reasons that the latency in *Goerli* and *Ropsten* is higher than in *Kovan* and *Rinkeby*.

Results of the system latency indicate the technical barrier on Ethereum-based system performance, which limits the usability of the proposed decentralised trust system to serve only small-scale services. Note that unlike the other test-nets, Ropsten performs *Proof-of-Work* consensus mechanism, similar with the Ethereum main-net, thus, it best reproduces the Ethereum production environment. As SCs, including *FeEx* and *Rep* SCs, are dedicated to performing critical tasks with minimal storage and computation, the performance of a DApp is heavily dependent on the BC platform but not the application built on top. In addition to Ethereum test-nets, most of permissionless BC main-nets, at this stage, offer limited performance in terms of both throughput and/or scalability. For instance, Bitcoin and Ethereum main-net only handle about 7 and 15 transactions per second[17]). As a consequence, the system performance immensely relies on an underlying BC network which requires further research on consensus mechanisms [31], off-chain [32] and sharding solutions [33], etc. for a better DApp ecosystem.

Besides the underlying BC platform, a variety of factors should be taken into account when deploying the trust system into real-world usages. For instance, *gas* cost for SC execution in Ethereum Virtual Machine is high as such SCs requires high volume storage for the state variables, as well as numerous operations and local variable manipulations in *Sc* SC and the cost for using *Oraclize* service in *Rep* SC. This calls for further research on SC optimisation [34] and better off-chain storage and calculation solutions.

Another limitation of the proposed approach is that the reputation calculation needs to be executed off-chain due to the extremely-high cost of *Rep* SC. As a consequence, the system has to leverage the Oraclize solution in order to circumvent the Oracle problem, which might lead to the dependency upon a third-party (i.e., Oraclize provider) and breaks the philosophy of a fully decentralised system. This dependency can be mitigated if an Oraclize provider deploys its service under a decentralised fashion, which is the solution approach to be pursued [35].

6.2. System security

The advanced capability of BC platform plays a key role in providing a secure and trustworthy environment for DApps. Although current BC and SC technologies still pose both performance limitations and security threats, we assume that the decentralised nature of the BC ensures there is no adversary can corrupt the BC network and change the content of the ledgers as this would imply majority of the network's resources are

15 https://github.com/nguyentb/Decentralised_Trust_Eth_IPFS/tree/master/packages/performanceAnalysis

16 The number of blocks increase counted when a transaction is broadcasted to the network until it is confirmed (written in the latest block).

17 https://blockchain.info/charts/n-transactions

compromised. Besides, there is no adversary who can impersonate another entity as the public-key cryptography (e.g., Elliptic Curve Digital Signature Algorithm (ECDSA) used in Ethereum) cannot be forged.

Security threats in the proposed decentralised trust system are from typical reputation-related attacks such as Self-promoting, Slandering (good/bad mouthing), and Whitewashing [36]. In our system, to be able to provide feedback, an entity is required to make a transaction toward the counter-party which, as a result, yields some fees, at least the transaction fee. Therefore, such reputation attacks are minimised as they always comes at cost. Importantly, the proposed reputation mechanism itself can mitigate such reputation attacks. For instance, if a newly-created entity (thus its reputation value is minimal), makes a transaction, and then gives bad/good feedback toward a victim; the contribution of this feedback to the reputation value of the victim is minimal. This is because the reputation value of the victim is calculated based on both experience and reputation score of participants who transact with the victim (indicated in Eqs. (6) and (7)). Obviously, if an entity is high-reputed (thus, probably not malicious) then the contribution (to one's reputation) is huge. Our reputation mechanism shares the same characteristics to Page-rank algorithm in Google web-ranking engine [37] that it is not easy to increase the reputation ranking of an entity by creating a lot of new Experience relationships toward it.

Feedback is typically subjective, and low-quality feedback toward an entity results in the imprecision of the associated Experience relationship, and, consequently, the reputation and trust relationships of the entity. The nature of any feedback-based evaluation systems is that it is impossible to fully prevent from such low-quality feedback and reputation attacks. However, we believe our approach can well mitigate these behaviours, particularly with a large number of honest participants over corrupt ones.

7. Conclusion

In this paper, we have provided a comprehensive concept, system model and design of a decentralised trust system for DApps ecosystem along with detailed analysis, algorithms, and simulations actualise the *DER* trust model. Foremost, we have developed a proof-of-concept system implementing the *DER* trust model on top of the Ethereum permissionless BC. The trust system is then able to incorporate with the DDS service for supporting data owners to select trustworthy storage nodes.

We have also provided technical difficulties along with prospective solutions as well as the implementation reference in the development of the proposed decentralised trust system. Existing technical barriers are also outlined which need further efforts to be successfully solved. We believe our research significantly contributes to further activities on trust-related research areas and open some future research directions to strengthen a trustworthy DApp ecosystem.

CRediT authorship contribution statement

Nguyen Truong: Conceptualization, Methodology, Software, Formal analysis, Validation, Writing - original draft. **Gyu Myoung Lee:** Investigation, Conceptualization, Supervision, Writing - review & editing. **Kai Sun:** Project administration, Writing - review & editing. **Florian Guitton:** Software. **YiKe Guo:** Investigation, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research was supported by the HNA Research Centre for Future Data Ecosystems at Imperial College London, United Kingdom and the Innovative Medicines Initiative 2 IDEA-FAST project under grant agreement No 853981.

Appendix A

This appendix section is dedicated to solving Lemmas 4.1 and 4.2 in Section 4.

A.1. Increase model lemma

Lemma A.1. *The Increase model defined in Eq. (1) is (*) a monotonically increasing function and (**) asymptotic to* 1.

Proof. From Eqs. (1) and (2), with $max_{Exp} = 1$, we have:

$$Exp_t = Exp_{t-1} + (1 - Exp_{t-1})\vartheta_t\ \alpha \tag{10}$$

Subtracting both sides of Eq. (10) from 1:

$$\begin{aligned} 1 - Exp_t &= 1 - (Exp_{t-1} + (1 - Exp_{t-1})\vartheta_t\ \alpha) \\ &= (1 - Exp_{t-1})(1 - \vartheta_t\ \alpha) \\ &= (1 - Exp_{t-2})(1 - \vartheta_t\ \alpha)(1 - \vartheta_{t-1}\ \alpha) \\ &= \cdots \\ &= (1 - Exp_0)\prod_{i=1}^{t}(1 - \vartheta_i\ \alpha) \end{aligned} \tag{11}$$

As $0 < Exp_0 < 1$, $0 < \alpha < max_{Exp} = 1$, and $0 < \vartheta_i < 1\ \forall i$; from Eq. (11) we have $0 < Exp_t < 1\ \forall t$. Therefore, Exp_t function defined in Eq. (1) is increasing as the increment value between Exp_t and Exp_{t-1} is $\vartheta_t \times \Delta Exp_t$ where $\Delta Exp_t = \alpha(1 - Exp_{t-1}) > 0$. Hence, Lemma (*) is proven.

Furthermore, as Increase model is for cooperative transactions, meaning that $\vartheta_i \geq \theta_{co}$; $\forall i \in \{1, \ldots, t\}$; from Eq. (11) we have:

$$0 < 1 - Exp_t \leq (1 - Exp_0)(1 - \theta_{co}\ \alpha)^t \tag{12}$$

As θ_{co}, α, and Exp_0 are the three pre-defined parameters in the range $(0, 1)$; therefore:

$$\lim_{t\to\infty}(1 - Exp_0)(1 - \theta_{co}\ \alpha)^t = 0 \tag{13}$$

Applying the Squeeze theorem on (12) and (13), we then have:

$$\lim_{t\to\infty}(1 - Exp_t) = 0 \tag{14}$$

In other word, the monotonically increasing Exp_t function is *asymptotic to* 1; hence Lemma (**) is proven. □

A.2. Reputation model lemma

Lemma A.2. *The reputation vector Rep exists and is unique.*

Proof. According to Eq. (8), *Rep exists* and is *unique* if both Rep_{Pos} and Rep_{Neg} exist and are unique.

The positive experience $N \times N$ matrix PE is constituted as follows:

$$PE(i,j) = \begin{cases} Exp(i,j) & \text{if } Exp(j,i) \geq \theta \\ 0 & \text{if } Exp(j,i) < \theta \end{cases} \tag{15}$$

Let us constitute an $N \times N$ diagonal matrix $\mathcal{M}$ whose diagonal elements $m_i = C_{Pos}(i)$, $\forall i \in \{1, \ldots, N\}$ and a matrix $\mathcal{J}$ is a $N \times N$ all-ones matrix.

Based on Eq. (6), Rep_{Pos} can be represented in matrix notation as follows:

$$Rep_{Pos} = (\frac{1-d}{N} \times \mathcal{J} + d \times PE \times \mathcal{M}^{-1}) \times Rep_{Pos} \quad (16)$$

Let us define the A_{Pos} matrix as follows:

$$A_{Pos} = \frac{1-d}{N} \times \mathcal{J} + d \times PE \times \mathcal{M}^{-1} \quad (17)$$

Thus, Eq. (16) can be re-written:

$$Rep_{Pos} = A_{Pos} \times Rep_{Pos} \quad (18)$$

From Eq. (18), we can see that Rep_{Pos} is the *eigenvector* of matrix A_{Pos} with the *eigenvalue* $= 1$. Let us define a matrix $P = A^T_{Pos}$; thus $P^T = A_{Pos}$. Therefore, Eq. (18) can be re-written as follows:

$$Rep_{Pos} = P^T \times Rep_{Pos} \quad (19)$$

Eq. (19) implies that Rep_{Pos} is the *stationary distribution* of a *Markov* chain whose transition probability matrix is P. Let us constitute a discrete-time *Markov* chain with the transition probability matrix $P = A^T_{Pos}$ consisting of N states and the probability to move from state i to state j is $P(i, j)$. Note that $\forall i, j \in \{1, .., N\}$, we have:

$$P(i,j) = A^T_{Pos}(i,j) = A_{Pos}(j,i) = \frac{1-d}{N} + d \times \frac{PE(j,i)}{m(j)} \quad (20)$$

The Markov chain can then be constructed as follows:

$$P(i,j) = \begin{cases} \frac{1-d}{N} + d \times \frac{PE(j,i)}{m(j)} & \text{if } Exp(j,i) \geq \theta \\ 1 - (\frac{1-d}{N} + d \times \frac{PE(j,i)}{m(j)}) & \text{if } Exp(j,i) < \theta \end{cases} \quad (21)$$

where θ is the threshold to differentiate positive and negative experiences. This *Markov* chain is a model of *random surfer* with *random jumps* over the experience relationships directed graph $G(V, E)$ [38–40]. The graph $G(V, E)$ is strongly connected with no dangling nodes. This is because any two nodes (x, y) with no prior transaction is set $Exp(x, y) = 0$, implying that the edge weight is 0; it does not mean there is no connection. This random surfer Markov chain, apparently, is a weighted PageRank model; as a result, its *stationary distribution*, Rep_{Pos}, exists and is *unique* [39–41].

Similarly, Rep_{Neg} vector *exists* and is *unique*. Therefore, the overall reputation vector *Rep exists* and is *unique*. □

Appendix B

This appendix section depicts some pseudo-code used in Section 5. Note that the source-code of the software is located at Github repo.[18] Solidity source-code for Smart Contracts is under `/packages/ethereum-core` and the performance evaluation scripts are under `/packages/performanceAnalysis` in the repository.

B.1. Data structure and state variable in FeEx SC

```
struct FeExStrut {
  uint expValue;
  uint fbScore;
  bool perFlag;
}
mapping (address=>mapping (address=>FeExStrut))
  public FeExInfo;
```

[18] https://github.com/nguyentb/Decentralised_Trust_Eth_IPFS.git

B.2. State variable in Rep SC

```
  mapping (address => uint)
    public RepPosInfo;
  mapping (address => uint)
    public RepNegInfo;
```

B.3. Oracle API used in Rep SC

```
import "./provableAPI.sol";
contract Rep is usingProvable {
 function oraclizeQuery() {
  // make an Oraclize query to the service using URL
  oraclize_query("URL", RepCalService_API_URL);
 }

 function __callback(bytes32 _requestID, string _result) {
  // only Oraclize is permitted to invoke the function
  require (msg.sender == oraclize_cbAddress());

  // update RepPosInfo & RepNegInfo
  RepPosInfo[addr] = getRepPos(_result, addr);
  RepPosInfo[addr] = getRepNeg(_result, addr);
 }
}
```

B.4. FeEx SC is integrated into DDS SC

```
contract DDS {
 function ePayment(address _storageNode,
  unit _amount, string _datahash) {
  ...
  if (success) {
   //call FeEx using deployed address scAddr
   FeEx fe = FeEx(scAddr);
   fe.enFeedback(msg.sender, _storageNode,
   string _transID);
  }
 }
}
contract FeEx {
 function enFeedback(address _owner,
  address _storageNode, string _transID);
 function expCal(address _owner, uint _fbScore,
  address _storageNode, string _transID);
}
```

References

[1] V. Buterin, et al., A next-generation smart contract and decentralized application platform, in: White Paper 3, 2014.

[2] M. Ali, J. Nelson, R. Shea, M.J. Freedman, Blockstack: A global naming and storage system secured by blockchains, in: {USENIX} Annual Technical Conference, 2016, pp. 181–194.

[3] N. Hackius, M. Petersen, Blockchain in logistics and supply chain: trick or treat? in: Proceedings of the Hamburg International Conference of Logistics (HICL), epubli, 2017, pp. 3–18.

[4] K. Korpela, J. Hallikas, T. Dahlberg, Digital supply chain transformation toward blockchain integration, in: Proceedings of the 50th Hawaii International Conference on System Sciences, 2017.

[5] H. Shafagh, L. Burkhalter, A. Hithnawi, S. Duquennoy, Towards blockchain-based auditable storage and sharing of iot data, in: Proceedings of the 2017 on Cloud Computing Security Workshop, ACM, 2017, pp. 45–50.

[6] R. Li, T. Song, B. Mei, H. Li, X. Cheng, L. Sun, Blockchain for large-scale internet of things data storage and protection, IEEE Trans. Serv. Comput. (2018).

[7] D. Johnston, S.O. Yilmaz, J. Kandah, N. Bentenitis, F. Hashemi, R. Gross, S. Wilkinson, S. Mason, The general theory of decentralized applications - DApps, 2014.

[8] V. Buterin, DAOS, DACs, DAS and more: An incomplete terminology guide, Ethereum Blog 6 (2014) 2014.

[9] N.B. Truong, T.-W. Um, B. Zhou, G.M. Lee, Strengthening the blockchain-based internet of value with trust, in: 2018 IEEE International Conference on Communications (ICC), IEEE, 2018, pp. 1–7.

[10] R. Urena, G. Kou, Y. Dong, F. Chiclana, E. Herrera-Viedma, A review on trust propagation and opinion dynamics in social networks and group decision making frameworks, Inform. Sci. 478 (2019) 461–475.
[11] F. Almenárez, A. Marín, D. Díaz, A. Cortés, C. Campo, C. García-Rubio, Trust management for multimedia P2P applications in autonomic networking, Ad Hoc Netw. 9 (4) (2011) 687–697.
[12] Z. Yan, P. Zhang, A.V. Vasilakos, A survey on trust management for internet of things, J. Netw. Comput. Appl. 42 (2014) 120–134.
[13] N.B. Truong, T.-W. Um, B. Zhou, G.M. Lee, From personal experience to global reputation for trust evaluation in the social internet of things, in: GLOBECOM 2017-2017 IEEE Global Communications Conference, IEEE, 2017, pp. 1–7.
[14] N.B. Truong, G.M. Lee, T.-W. Um, M. Mackay, Trust evaluation mechanism for user recruitment in mobile crowd-sensing in the internet of things, IEEE Trans. Inf. Forensics Secur. 14 (10) (2019) 2705–2719.
[15] Z. Yang, K. Yang, L. Lei, K. Zheng, V.C. Leung, Blockchain-based decentralized trust management in vehicular networks, IEEE Internet Things J. 6 (2) (2018) 1495–1505.
[16] X. Chen, J. Ding, Z. Lu, A decentralized trust management system for intelligent transportation environments, IEEE Trans. Intell. Transp. Syst. (2020).
[17] A. Moinet, B. Darties, J.-L. Baril, Blockchain based trust & authentication for decentralized sensor networks, 2017, arXiv preprint arXiv:1706.01730.
[18] W. She, Q. Liu, Z. Tian, J.-S. Chen, B. Wang, W. Liu, Blockchain trust model for malicious node detection in wireless sensor networks, IEEE Access 7 (2019) 38947–38956.
[19] M. Debe, K. Salah, M.H.U. Rehman, D. Svetinovic, IoT Public fog nodes reputation system: A decentralized solution using ethereum blockchain, IEEE Access 7 (2019) 178082–178093.
[20] P. Kochovski, S. Gec, V. Stankovski, M. Bajec, P.D. Drobintsev, Trust management in a blockchain based fog computing platform with trustless smart oracles, Future Gener. Comput. Syst. 101 (2019) 747–759.
[21] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things, in: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, 2012, pp. 13–16.
[22] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G.M. Voelker, S. Savage, A fistful of bitcoins: characterizing payments among men with no names, in: Proceedings of the 2013 Conference on Internet Measurement Conference, 2013, pp. 127–140.
[23] N. Tyagi, S. Sharma, Weighted page rank algorithm based on number of visits of links of web page, Int. J. Soft Comput. Eng. (2012) 2231–2307.
[24] S. Brin, L. Page, Reprint of: The anatomy of a large-scale hypertextual web search engine, Comput. Netw. 56 (18) (2012) 3825–3833.
[25] G. Jawaheer, P. Weller, P. Kostkova, Modeling user preferences in recommender systems: A classification framework for explicit and implicit user feedback, ACM Trans. Interact. Intell. Syst. 4 (2) (2014) 1–26.
[26] S.G. Roberts, R.I. Dunbar, T.V. Pollet, T. Kuppens, Exploring variation in active network size: Constraints and ego characteristics, Social Networks 31 (2) (2009) 138–146.
[27] W. Xing, A. Ghorbani, Weighted pagerank algorithm, in: Proceedings. Second Annual Conference on Communication Networks and Services Research, 2004, IEEE, 2004, pp. 305–314.
[28] A. Arasu, J. Novak, A. Tomkins, J. Tomlin, PageRank computation and the structure of the web: Experiments and algorithms, in: Proceedings of the Eleventh International World Wide Web Conference, Poster Track, 2002, pp. 107–117.
[29] S.D. Kamvar, T.H. Haveliwala, C.D. Manning, G.H. Golub, Extrapolation methods for accelerating PageRank computations, in: Proceedings of the 12th International Conference on World Wide Web, 2003, pp. 261–270.
[30] X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A.B. Tran, S. Chen, The blockchain as a software connector, in: 2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA), IEEE, 2016, pp. 182–191.
[31] Z. Zheng, S. Xie, H. Dai, X. Chen, H. Wang, An overview of blockchain technology: Architecture, consensus, and future trends, in: 2017 IEEE International Congress on Big Data (BigData Congress), IEEE, 2017, pp. 557–564.
[32] J. Poon, T. Dryja, The bitcoin lightning network: Scalable off-chain instant payments, 2016.
[33] M. Zamani, M. Movahedi, M. Raykova, Rapidchain: Scaling blockchain via full sharding, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018, pp. 931–948.
[34] T. Chen, X. Li, X. Luo, X. Zhang, Under-optimized smart contracts devour your money, in: 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), IEEE, 2017, pp. 442–446.
[35] J. Adler, R. Berryhill, A. Veneris, Z. Poulos, N. Veira, A. Kastania, Astraea: A decentralized blockchain oracle, in: 2018 IEEE International Conference on Internet of Things (IThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), IEEE, 2018, pp. 1145–1152.
[36] K. Hoffman, D. Zage, C. Nita-Rotaru, A survey of attack and defense techniques for reputation systems, ACM Comput. Surv. 42 (1) (2009) 1–31.
[37] K. Avrachenkov, N. Litvak, The effect of new links on google pagerank, Stoch. Models 22 (2) (2006) 319–331.
[38] L. Page, S. Brin, R. Motwani, T. Winograd, The PageRank citation ranking: Bringing order to the web, Technical Report, Stanford InfoLab, 1999.
[39] A. Blum, T.H. Chan, M.R. Rwebangira, A random-surfer web-graph model, in: 2006 Proceedings of the Third Workshop on Analytic Algorithmics and Combinatorics (ANALCO), SIAM, 2006, pp. 238–246.
[40] P. Chebolu, P. Melsted, Pagerank and the random surfer model, in: SODA, 2008, pp. 1010–1018.
[41] T. Haveliwala, S. Kamvar, G. Jeh, An analytical comparison of approaches to personalizing pagerank, Technical Report, Stanford, 2003.

Dr. Nguyen B. Truong is currently a Research Associate at Data Science Institute, Imperial College London, United Kingdom. He received his Ph.D, MSc, and BSc degrees from Liverpool John Moores University, United Kingdom, Pohang University of Science and Technology, Korea, and Hanoi University of Science and Technology, Vietnam in 2018, 2013, and 2008, respectively. He was a Software Engineer at DASAN Networks, a leading company on Networking Products and Services in South Korea in 2012–2015. His research interest is including, but not limited to, Data Privacy, Security, and Trust, Personal Data Management, Distributed Systems, and Blockchain.

Dr. Gyu Myoung Lee received his BS degree from Hong Ik University and MS, and PhD degrees from the Korea Advanced Institute of Science and Technology (KAIST), Korea, in 1999, 2000 and 2007, respectively. He is a Professor at Department of Computer Science, Liverpool John Moores University, UK. He is also with KAIST as an adjunct professor. His research interests include Future Networks, IoT, and multimedia services. He has actively contributed to standardisation in ITU-T as a Rapporteur, oneM2M and IETF. He is chair of the ITU-T Focus Group on data processing and management to support IoT and Smart Cities & Communities.

Dr. Kai Sun is the Operation Manager of the Data Science Institute at Imperial College London. She received the MSc degree and the Ph.D degree in Computing from Imperial College London, in 2010 and 2014, respectively. From 2014 to 2017, she was a Research Associate at the Data Science Institute at Imperial College London, working on EU IMI projects including U-BIOPRED and eTRIKS, responsible for translational data management and analysis. She was the manager of the HNA Centre of Future Data Ecosystem in 2017–2018. Her research interests include translational research management, network analysis and decentralised systems.

Mr. Florian Guitton received a BSc in Software Engineering from Epitech (France) in 2011 and a MSc in Advanced Computing from the University of Kent (United Kingdom) in 2012. In 2012 he joined the Discovery Sciences Group at Imperial College London where he became Research Assistant working on iHealth, eTRIKS and IDEA-FAST EU programs. He is currently a PhD candidate at Data Science Institute, Imperial College London working on distributed data collection and analysis pipeline in mixed-security environments with the angle of optimising user facing experiences.

Dr. Yike Guo (FREng, MAE) is the director of the Data Science Institute at Imperial College London and the Vice-President (Research and Development) of Hong Kong Baptist University. He received the BSc degree in Computing Science from Tsinghua University, China, in 1985 and received the Ph.D in Computational Logic from Imperial College London in 1993. He is a Professor of Computing Science in the Department of Computing at Imperial College London since 2002. He is a fellow of the Royal Academy of Engineering and a member of the Academia Europaea. His research interests are in the areas of data mining for large-scale scientific applications including distributed data mining methods, machine learning and informatics systems.