Journal of Parallel and Distributed Computing 143 (2020) 167–178

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

BPS: A reliable and efficient pub/sub communication model with blockchain-enhanced paradigm in multi-tenant edge cloud

Bobo Huang [a], Rui Zhang [a], Zhihui Lu [a,b,*], Yiming Zhang [a], Jie Wu [a,e,*], Lu Zhan [c], Patrick C.K. Hung [d]

[a] School of Computer Science, Fudan University, Shanghai, China
[b] Shanghai Blockchain Engineering Research Center, Shanghai, China
[c] School of Economics, Fudan University, Shanghai, China
[d] Faculty of Business and IT, University of Ontario Institute of Technology, Canada
[e] Engineering Research Center of Cyber Security Auditing and Monitoring, Ministry of Education, Shanghai, China

ARTICLE INFO

Article history:
Received 28 February 2020
Received in revised form 4 May 2020
Accepted 13 May 2020
Available online 21 May 2020

Keywords:
Blockchain
Smart city
Pub/sub system
Edge cloud
Security

ABSTRACT

In recent years, with the rapid development of smart city, prevalent pub/sub (publish/subscribe) streaming systems have been increasingly employed as upstream middleware layer in multi-tenant edge clouds, and feed large volume of data gathered from IoT devices of different tenants into downstream systems (e.g., data analytics and warehouse). A shared tenancy model where multiple untrusted applications or tenants utilize the same pub/sub system is generally exploited in edge cloud, which poses crucial challenges including privacy-sensitive data/metadata access threat and critical metadata modification by unauthorized tenants. A centralized monitoring node is invariably adopted in existing security strategies (such as ACL, TLS), which causes the pub/sub streaming model vulnerable to external malicious attacks and single point failure.

In this paper, inspired by outstanding features of blockchain including tamper-resistance, decentralization, strong consistency, and traceability, we propose BPS, a general and decentralized Blockchain-enhanced Pub/Sub communication model for multi-tenant edge cloud, to redesign pub/sub system internal security mechanisms. Specifically, by exploiting blockchain technology, BPS can detect the illegal operations and behaviors from both malicious tenants and untrusted publishers or subscribers. BPS directly leverages Merkel Hash Tree (MHT) of blockchain to verify the integrity of critical and confidential metadata. Regarding authorization, BPS introduces smart-contract-enabled fine-grained control over partition topic-classified messages by storing access control list (ACL) into an append-only blockchain ledger. Additionally, an incentive mechanism is employed in BPS to reward honest publishers and subscribers. We implement BPS prototype based on Kafka and EoS blockchain. Our security analysis and extensive experiments demonstrate that BPS outperforms the state-of-the-art pub/sub streaming system Kafka in security with minimal performance overhead.

1. Introduction

In the past few decades, with the massive deployment of IoT infrastructure in smart city, an increasing amount of data is generated and gathered by a wide variety of IoT devices or intelligent terminals belonging to various enterprises or organizations, such as IoT sensors, mobile phones, smart cameras. Generally, large-scale distributed big data processing pipelines are required over such massive IoT datasets to explore the potential value. Due to constrained resources of general IoT devices, many data-intensive computing tasks like pub/sub (publish/subscribe) streaming processing have to be offloaded into the emerging edge cloud shared by multiple tenants. Edge cloud computing dramatically decreases transfer latency over large amounts of IoT datasets by deploying computing and storage services at the edge of the network, which bridges the data transmission gap between edge devices and core cloud.

On the other hand, many cloud service providers have deployed pub/sub services closer to edge devices in edge cloud centers [1,18,21,24]. Specifically, most pub/sub streaming systems like Kafka [13,25], ZeroMQ [44], RabbitMQ [33] and Pulsar [4] act as upstream middleware layer enabling downstream systems (e.g., Spark, Flink, Federated learning, Tensorflow) to share

* Corresponding authors at: School of Computer Science, Fudan University, Shanghai, China.

E-mail addresses: huangbb16@fudan.edu.cn (B. Huang), zhangrui19@fudan.edu.cn (R. Zhang), lzh@fudan.edu.cn (Z. Lu), zhangym19@fudan.edu.cn (Y. Zhang), jwu@fudan.edu.cn (J. Wu), zhanlu@fudan.edu.cn (L. Zhan), patrick.hung@uoit.ca (P.C.K. Hung).

https://doi.org/10.1016/j.jpdc.2020.05.005
0743-7315/© 2020 Elsevier Inc. All rights reserved.

data feeds with each other via subscribing to topics of interest [23]. As a typical implementation of pub/sub streaming systems, Kafka employs topic-based and pull-enabled producer–broker–consumer model for high throughput and high bandwidth. With a standard Kafka system setup, any application or user can produce any messages into any topics, as well as consume data from any topics. In an edge cloud with a shared tenancy model where multiple untrusted applications and tenants employ the same Kafka cluster, the features of untrusted multiple tenants result in crucial security challenges including privacy-sensitive data/metadata access threat and confidential metadata modification by unauthorized tenants in terms of authentication and authorization among different tenants. In this paper, we focus on the security of topic-based and broker-enabled pub/sub streaming systems.

Most existing topic-based broker-enabled pub/sub models (especially like Kafka [25]) for edge cloud depend on centralized cloud servers (e.g., ZooKeeper) to maintain and manage critical metadata and access control list (ACL). For instance, Kafka relies on centralized ZooKeeper servers to manage load balancing of producers and consumers, registration of the broker/consumer/topic, and message consuming offset.

Despite the high performance, the centralized model under a shared tenancy model results in inevitable security threats including single point failure and key metadata tampering. Specifically, publishing or subscribing message services provided by edge cloud will not be available once the centralized pub/sub models suffer from DDoS attacks. If ACL is modified by corrupted brokers or malicious third parties, unauthorized producers owned by untrusted tenants can produce spam messages to unauthorized topics owned by other honest tenants. Meanwhile, unauthorized consumers can consume privacy-sensitive topic messages owned by other tenants. Thus, confidentiality, anonymity and data integrity of honest tenants cannot be guaranteed. Besides, untrusted cloud servers may commit misbehaviors of storing tenant's outsourced data. And the traditional broker-based pub/sub models lack an incentive or penalty mechanism for publisher and subscriber behaviors. The loose coupling between publishers and subscribers poses further challenges to mature security strategies. How to enable malicious behaviors detection, reliable authorization, confidential data/metadata integrity verification in a decentralized manner for topic-based broker-enabled pub/sub streaming systems in the untrusted multi-tenant edge cloud are critical challenges.

Meanwhile, as an emerging reliable and decentralized storage technology, blockchain has gained tremendous attention from both industry and academia. A blockchain is an immutable append-only database, where each block with many signed transactions is linked with the previous block by using a hash pointer. All blocks on blockchain are managed and replicated in a decentralized manner via distributed consensus protocol across untrusted participants. Once data packed by transactions is appended into blockchain, almost nobody can modify it. To accommodate the demand for various application scenarios, Turing-complete smart contract has been supported in mainstream blockchain implementations [2,11,31,40]. Due to the unique features of decentralization, tamper-resistance, consensus protocol, and traceability, blockchain has been widely leveraged in many fields (e.g., finance [31], IoT [14]) to enhance security level of different applications [7,9,26,32,39,42,45,47], which is called the blockchain-enhanced paradigm. For instance, Chenhan et al. [42] employed tamper-resistance and non-repudiation features of blockchain to address the distrust issues of big data sharing in edges. GEM2-Tree in [45] explored authenticated range queries over a hybrid-storage blockchain.

Inspired by blockchain-enhanced paradigm, we propose BPS, a general blockchain-enhanced pub/sub communication model for distrusted multi-tenant edge cloud, which redesigns the security mechanism of topic-based and broker-based pub/sub streaming system via closely coupling blockchain properties. We aim to enable malicious behaviors detection, reliable authorization, confidential data/metadata integrity verification with minimal performance overhead in BPS. To achieve such goals, we determine three critical design choices. First, considering huge performance overhead brought by directly storing raw topic-classified messages from different tenants into blockchain, we exploit a hybrid storage architecture where only key metadata (like ACL, identity, partition offsets of consumers, reputation value, consumer group, topic and partition information, etc.) are stored on-chain and raw messages are still persisted to log-based store located at brokers. Meanwhile, a hash for each raw message is maintained on-chain and treated as the proof of the raw messages. The on-chain hashes are employed to authenticate the messages retrieved from off-chain log store to ensure data integrity. Second, we append operation logs (oplogs) like message *Produce* and *Fetch* operations into on-chain, which prevents malicious publishers or subscribers from covering up misbehaviors by changing oplogs. Taking the cost of committing oplogs to blockchain into consideration, we adopt a batched-actions committing scheme to improve throughput of oplogs committing. Third, we develop an ACL-enabled smart contract to determine whether the requests-corresponding publishers or subscribers are authorized to produce or consume on specific topics. Due to the high latency of querying ACL over smart contract, we cache hot ACL entries with an expiration time in brokers. Additionally, an incentive mechanism is exploited to reward honest clients and punish the malicious ones. Compared to the start-of-the-art Kafka system, BPS achieves the expected reliability and security with at least *0.6%* loss in *Produce* throughput and at least *4.5%* loss in *Fetch* throughput under millions of request workloads.

To the best of our knowledge, BPS is the first proposal that provides a secure, privacy-protected and edge-cloud-oriented broker-based pub/sub streaming system in a multi-tenant scenario with blockchain-enhanced paradigm. The critical contributions of this paper are summarized as follows.

- By giving a detailed security analysis and threat model of multi-tenant-shared pub/sub streaming model, we find crucial challenges in terms of centralized authentication, authorization and data integrity.
- To accommodate the demand for higher security level required by shared tenancy model, we propose a general and decentralized blockchain-enhanced topic-based pub/sub streaming model called BPS.
- We detect key metadata corruption in a timely manner by exploiting an MHT-enabled metadata integrity verification.
- A smart contract-enabled fine-grained access control over topics is utilized to protect critical metadata against being tampered by untrusted tenants. Additionally, we put forward an incentive mechanism to encourage publishers/subscribers to collaborate honestly with BPS.
- Based on Kafka and EoS Blockchain, we implement a proof-of-concept prototype, providing a detailed security analysis of the expected BPS scheme, and conduct extensive experiments.

The structure of the paper is organized as follows. In Section 2, we investigate the related works on Pub/Sub streaming systems and blockchain, then discuss threat model and our motivation. Section 3 outlines the overview of BPS including design goals, architecture, and challenges. The technical details of Blockchain enhanced Pub/Sub system model and implementation are described in Section 4. Section 5 gives the security analysis and conducts extensive experiments to demonstrate the performance of our proposal. Finally, Section 6 concludes this paper.

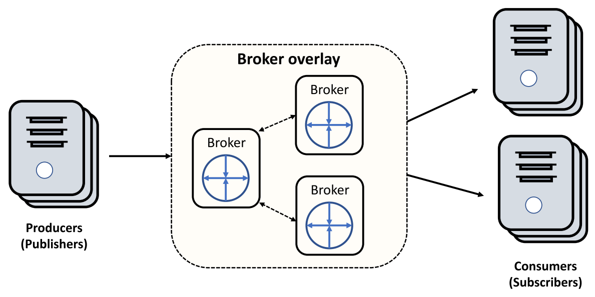


Fig. 1. The architecture of generic broker-based pub/sub systems.

2. Related work & threat model

In this section, we first review the related works of security schemes on topic-based and broker-enabled pub/sub systems. Next, we introduce blockchain characteristics and discuss the blockchain enhanced security paradigm. Finally, the threat model of our work is demonstrated.

2.1. Topic-based and broker-enabled pub/sub systems

In general, pub/sub systems can be categorized into topic- or content-based, broker-enabled or broker-less. Publishers and subscribers are loosely coupled in the broker-based model. A generic architecture of broker-based pub/sub systems is shown in Fig. 1. Topic-based pub/sub data model generally associates messages with different topics and delivers them to all interested subscribers. Particularly, in a broker-enabled pub/sub system, the messages delivery from brokers to subscribers can be summarized as push- (like RabbitMQ [33]) and pull-based (like Kafka [25]). In this paper, we mainly focus on topic-based and broker-enabled pub/sub systems with a pull-based model.

2.1.1. Shared pub/sub model in multi-tenant edge cloud

With the rapid growth of IoT and edge cloud computing, increasingly massive scale of data produced by various edge devices from different tenants needs to be offloaded into shared pub/sub systems in edge cloud owned by service providers. LinkedIn, for example, reports that nearly two Petabytes of data every week needs to be processed by pub/sub systems [24]. In the shared pub/sub model in multi-tenant edge cloud, the middle brokers cluster is leveraged to (1) handle *Produce* requests from upstream IoT edge nodes (as publishers) owned by different untrusted tenants, (2) persist messages data in partitioned logs and maintain critical metadata, (3) and response to *Fetch* requests from the downstream untrusted streaming processing services (as subscribers) employed by corresponding tenants. In the untrusted multi-tenant environment, we assume that tenants, upstream publishers, downstream subscribers, and broker cloud servers are unreliable and dishonest. Therefore, effective security and privacy policies in the shared multi-tenant pub/sub model are urgently demanded. The major security requirements are as follows:

- **Misbehaviors Monitoring**: Need to ensure that any illegal or malicious operations from unauthorized tenants, corrupted broker servers, and dishonest publishers or subscribers can be traced and detected.
- **Data Integrity**: Confidential metadata and critical portioned messages should be protected from being illegally tampered by malicious tenants or corrupted brokers. Moreover, the integrity of the data/metadata can be verified in an efficient manner.
- **Multi-tenant Friendly Authorization**: Message data isolation between different tenants is achieved by topic-enabled grouping. That is to say, a publisher owned by one specific tenant can only produce messages of the assigned topics, while a subscriber can only subscribe to assigned topics. The topics are only visible to authorized publishers and subscribers owned by untrusted tenants. A strict and fine-grained access control over topic-classified messages should be performed in a multi-tenant friendly manner.

Existing research has focused on resource orchestration optimization for multi-tenant edge cloud to improve resource utilization and performance [8,27]. So far, little attention has been paid to the shared (pub/sub) service security issues introduced by multi-tenant edge cloud.

2.1.2. Security for pub/sub systems

A few previous works focused on enhancing the security of content-based broker-less pub/sub systems [3,29,30,36–38]. For instance, Anusree et al. [3] employed Elliptic Curve identity-based signcryption [16] to provide unforgeability, confidentiality, and forward secrecy required by broker-less pub/sub service. The proposal in [38] leveraged hierarchical identity-based encryption to realize security mechanisms for broker-less pub/sub system. Muhammad et al. [36,37] proposed the pairing-based cryptography mechanism and fine-grained key management to guarantee authentication and confidentiality of a broker-less content-based pub/sub system, which requires a trusted centralized master server to maintain privacy-sensitive keys.

Additionally, there are other prior research focusing on authentication and privacy protection in broker-enabled or cloud-based pub/sub services [12,22,43]. Specifically, the scheme in [22] realized confidentiality and reliable access control for pub/sub systems with untrusted brokers through combining attribute-based encryption with searchable encryption. Regarding privacy preserving in cloud platforms, an attributed-keyword based pub/sub scheme called AKPS in [43] was proposed to preserve published messages against unauthorized cloud servers. Besides, a shell game algorithm-based probabilistic forwarding in Anon-PubSub [12] was employed to guarantee the anonymity of participants in pub/sub systems.

Generally, traditional security mechanisms for pub/sub model reply on centralized master nodes to provide authentication and confidentiality, which is vulnerable to single point failure. Furthermore, past pub/sub systems lack sufficient considerations for the isolation security issues caused by multi-tenant contention in shared edge cloud. To mitigate the above security bottlenecks and accommodate the demand for security in shared edge cloud, we implement a multi-tenant friendly pub/sub model in a completely decentralized manner. Table 1 summarizes the significant differences between BPS and typical pub/sub systems.

2.2. Blockchain technology

With the rapid evolution of blockchain [2,11,31,40,48], more efficient consensus algorithms are employed, which result in lower consensus latency and higher transaction throughput, as shown in Table 2. Meanwhile, many research efforts [14,17,19,20] are dedicated to optimizing the internal mechanism of blockchain systems. Considering the demand for high performance, we employ the emerging EoS [11] with permissioned mode as the underlying blockchain of BPS. The generic blockchain framework is shown in Fig. 2.

Table 1
Comparison with previous pub/sub systems.

Property	Broker-less & Content-based pub/sub				Broker-enabled & Topic-based pub/sub					
	Tariq [37]	Anusree [3]	Shitole [34]	AnonPubSub [12]	Ion [22]	Lv [28]	AKPS [43]	SPS [46]	Kafka [25]	BPS
Confidentiality	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Anonymity	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓
Authorization	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Performance	✓	✓	✓	✗	✗	✗	✓	✗	✓	✓
Integrity verification	✓	✓	✓	✓	✗	✓	✗	✓	✗	✓
Multi-tenant isolation	✗	✗	✗	✗	✗	✗	✓	✗	✓	✓
Tamper-resistance	✗	✗	✗	✗	✗	✓	✗	✓	✗	✓
Decentralized security	✗	✗	✗	✗	✗	✓	✗	✓	✗	✓

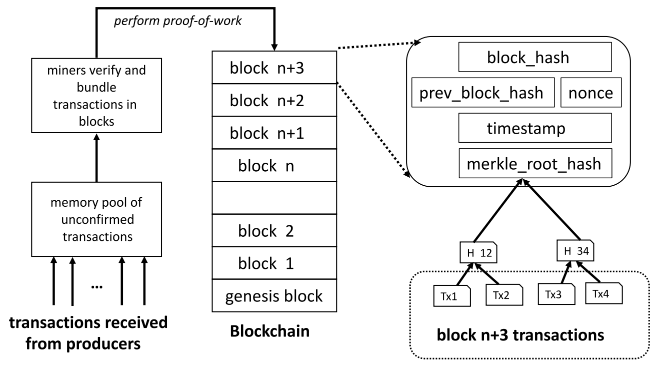


Fig. 2. The generic architecture of blockchain platform.

Table 2
Comparison of typical blockchain systems.

Platform	Block interval	Consensus	TPS
Bitcoin	10 min	PoW [31]	About 7
Ethereum	10 to 20 s	PoS [40]	<100
Fabric	3 to 6 s	PBFT [2]	>1000
EoS	0.5 s	DPoS [11]	million

2.2.1. Blockchain enhanced paradigm

Due to the security features and programmability described above, blockchain has aroused massive attention in academic and industrial fields [6,7,10,15,26,41,42]. BlockchainDB [15] proposed a database layer over blockchains by integrating classical data management and standard query interfaces, which improves the performance and scalability of blockchain-enabled data sharing. Certchain in [7] leveraged decentralized blockchain to enable public and efficient certificate audits for TLS connections, which can eliminate the safety bottlenecks induced by traditional centralized revoked certificates checking schemes. CrowdBC in [26] realized a blockchain-enabled decentralized crowdsourcing framework to replace the existing centralized trust-based model, which protects users' privacy with low transaction fees.

Motivated by the blockchain-enhanced paradigm, we mainly explore how to employ decentralized blockchain to mitigate the security bottlenecks discussed in Section 2.1.1.

2.2.2. Characteristics on blockchain for BPS

A typical blockchain has the following powerful characteristics to enable higher security level for BPS:

Decentralization: Blockchain employs a distributed consensus protocol to guarantee the consistency of on-chain data across all participants, instead of relying on trusted third parties. Such decentralized property of blockchain can naturally fight against attacks like DoS. BPS also leverages such a decentralization model to replace the traditional centralized pub/sub model.

Tamper-Resistance: All transactions carrying data are recorded blockchain ledger. Once data is appended to blockchain, almost nobody can modify or tamper with it. A secure and reliable on-chain storage for key data/metadata in BPS is ensured by this feature.

Smart Contract: Smart contract is a trusted Turing-complete program running atop of blockchain. The consensus protocol of blockchain provides execution integrity for smart contracts. In BPS, smart contracts for ACL, topics, and other key metadata are developed to ensure authenticity.

Anonymity: A few inner mechanisms are proposed to ensure the anonymity of on-chain membership transactions [5,14,35]. Due to these mechanisms, the identity privacy of blockchain users can be preserved and avoid being leaked. BPS employs this feature to guarantee the anonymity of producers' and consumers' identities.

Merkel Hash Tree: Blockchain constructs Merkle Hash Tree (MHT) for every block by pairing and hashing the transactions until one single hash (called Merkle Root) remains. So, the integrity of BPS metadata/data can be verified through on-chain MHT.

2.3. Threat model

In multi-tenant edge cloud, an adversary (e.g., a malicious publisher, subscriber, or tenant) generally attempts to attack a generic broker-based pub/sub system for three goals: (1) to insert, delete, or tamper with the topic-related ACL operations for making clients' *Produce* or *Fetch* requests validation failed; (2) to tamper with the partition offsets of consumers by attacking centralized master nodes such as Zookeeper servers; (3) to eavesdrop, forge and tamper with data messages belonging to honest tenants for compromising data integrity without being detected. Additionally, we make several general cryptographic assumptions. More specifically, an attacker cannot control more than 51% participate nodes under the blockchain-enhanced paradigm while signatures cannot be forged without corresponding private keys.

3. Overview

There are three kinds of critical entities in our BPS system over multi-tenant edge cloud: producer, consumer and broker, as shown in Fig. 3. A producer belonging to one specific tenant is the upstream entity who intends to establish connections with brokers and publish messages over tenants' corresponding topics into partitions in brokers. A consumer owned by one specific tenant can belong to a specific consumer group while acting as the downstream entity required to handshake with all brokers, subscribe to tenant-corresponding topics, and pull authorized messages from brokers. Generally, one tenant can have multiple producers and consumers corresponding to specific topics, and can pay the service provider for using computing and storage resources on brokers. Multi-tenancy is enabled by specifying which topics each tenant can produce or consume. Brokers are shared among various tenants while treated as messages

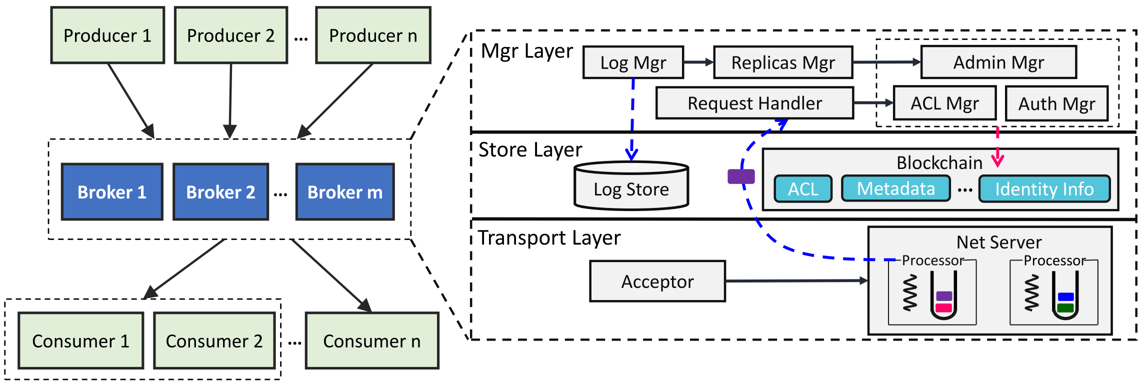


Fig. 3. The overview of BPS architecture.

forwarder between producers and consumers. Brokers handle the publish-related requests with specific topics from producers and check whether the producers have write permission for the topics. Meanwhile, brokers handle with the subscribe-related requests from consumers and determining whether the consumers have read permission for the specific topics. Instead of traditional centralized schemes, BPS maintains the blockchain on all broker nodes, and manages critical metadata (including ACL, topics, partitions, partition offsets and membership) on blockchain-bound brokers in a decentralized fashion. To support transparent audit, we store all privacy-sensitive operations over topics into on-chain. Specifically, by querying the blockchain, a manager of the edge cloud can detect whether there are illegal data access, malicious operations and misbehaviors from corrupted brokers or malicious publishers and subscribers. It is worth noting that the blockchain of BPS works in permissioned mode, which means only authorized nodes can participate in key metadata management.

3.1. *Design goals*

In order to achieve a safe and efficient pub/sub streaming service in a shared tenancy model, we need to determine a trade-off between security, reliability and high performance. That is to say that, we must ensure reliable confidentiality, anonymity, authentication, and authorization across multiple tenants, while minimizing performance overhead to provide relatively high message throughput and low handling latency. The specific goals are as follows:

- **Data integrity.** The integrity of messages and key metadata should be ensured by the necessary verifications and audits.
- **High query efficiency.** All operations (like register, query, publish or subscribe) on a specific topic can be tracked and audited without having to traverse the entire blockchain.
- **Intrusion tolerance.** Even if a broker crashes due to a malicious attack (such as DoS), publishers and subscribers served by this broker can be taken over by other brokers, and the metadata or messages partitions stored on the broker have reliable copies on other brokers.

3.2. *BPS architecture: Blockchain-enhanced pub/sub model*

In this paper, we design a blockchain-enhanced topic-based pub/sub streaming system which consists of three primary components: *broker*, *producer*, and *consumer*. In a multi-tenant shared edge cloud for smart cities, *producer* usually represents upstream edge applications located at edge nodes/devices (such as smart medical devices, smart shared bicycles, and smart homes) owned by untrusted tenants. Large amounts of IoT datasets are collected or generated by producers, which leads to the need for further distributed big data processing pipelines for tenants (also called data owners). *Consumer* corresponds to downstream streaming processing system (like Spark, Tensorflow) employed by tenants to data-intensive mining and inference computation. As the core component of BPS, *broker* manages the message data from upstream in the form of topics, while passively forwarding the topic-classified messages to the corresponding downstream consumers employed by tenants. In other words, producers publish topic-specified messages to brokers in a push-based manner (*Produce*) while consumers subscribe assigned topics and consume messages in a pull-based style (*Fetch*). Message data isolation between various tenants is achieved through topic-based grouping and blockchain-enabled fine-grained access control. This paper focuses on the design and descriptions of the core *broker* component. The overview of BPS architecture is exhibited in Fig. 3.

All broker nodes in BPS build up a blockchain network with a permissioned mode. A broker is also a core process running on the BPS server which is composed of *broker manager layer*, *store layer*, and *transport layer*. In *broker manager layer*, we propose blockchain-enhanced fine-grain access control to authorize the requests received from producers and consumers. To replay and trace the history topic operations, we define a new data structure called *TopicOper* to represent topic operations, which is recorded in blockchain in the form of Merkle Hash Tree (MHT). In *store layer*, trusted storage and computing services are provided due to the tamper-resistance and consensus properties of blockchain. And privacy-sensitive data can be stored on-chain. In *transport layer*, various messages from publishers and subscribers are received, deserialized, security checked and forwarded to broker manager layer for handling. This paper focuses on the broker component. We will give a detailed description to these three layers of broker.

Transport Layer. This layer consists of two core components: *Acceptor* and *Net Server*. The acceptor is mainly responsible for listening to the channel connection request and distributing it to the *Processor* in the *Net Server* in a round-robin manner. A *Net Server* contains multiple processors, each processor corresponds to a response queue, and all processors share the same global request queue. A processor is generally used to deserialize the requests and push them to the request queue, while serializing the responses and forward them to the remote client. Additionally, encryption and decryption of in-flight data using SSL/TLS is also performed at the transport layer.

Store Layer. As a trusted storage scheme, blockchain can generally store text, documents and images. However, considering that messages in BPS are usually large, directly storing raw data to the blockchain limits the scalability of BPS brokers. To tackle this problem, we adopt a hybrid storage architecture, in which only

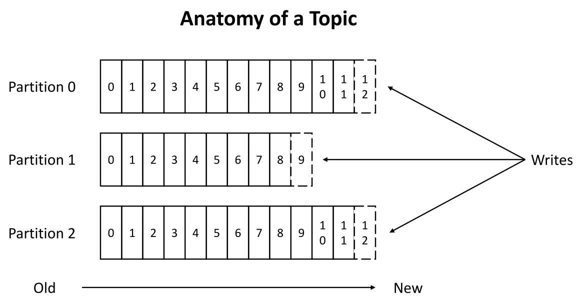


Fig. 4. The process of appending messages to partition.

small security-sensitive metadata like *ACL*, *identity information* and *topic partitions metadata* are appended to *Blockchain* and data messages are persisted to corresponding partitions in *Log Store* (as shown in Fig. 4). Meanwhile, every topic message has a corresponding cryptographic hash which is maintained on-chain as witness of the raw object.

Broker Manager Layer. *Broker mgr layer* is mainly composed of *Request Handler*, replicas manager (*Replicas Mgr*), persistence manager (*Log Mgr*), admin manager (*Admin Mgr*), ACL manager (*ACL Mgr*) and authentication manager (*Auth Mgr*). Through invoking corresponding smart contract programs executed on blockchain, admin manager primarily maintains the operations over core meta-data such as *TopicOper*, topic, partition, replicas, consumer offsets and membership. *ACL Mgr* is primarily responsible for registration, update and validation of the permissions of producers and consumers on specific topics. Each ACL entry is defined in the classical format of *"Principal P is [Allowed/Denied] Operation O From Host H on any Resource R matching ResourcePattern RP"*. *Auth Mgr* is used to manage the on-chain identity information of the producer, consumer and broker. *Replicas Mgr* which consists of a set of replicas is leveraged to manage all partitions on the local broker and replicas synchronization with other brokers. More specifically, when authorized messages need to be persisted to *Log Store*, *Replicas Mgr* gets the target partition instance based on the input *partitionId* and append messages to partition header in the form of logs. *Log Mgr* is mainly responsible for creating, retrieving, and cleaning up logs.

The request handler can receive and handle requests from the request queue of the transport layer and push the processing result into response queue. To be specific, when the request handler receives an operation request from one authenticated producer or consumer, it should determine whether or not the particular producer or consumer is allowed to write or read on specific topics by utilizing *ACL Mgr* to check on-chain ACL. If the operation over topic is invalid, a topic operation log with description such as operator name, topic name, operation type, and valid status is generated and recorded into blockchain by *Admin Mgr*. Otherwise, the authorized operation over some topics is performed and a generated valid topic operation is stored on-chain. For example, when a *Fetch* request from specific consumer is received, the request handler first employs on-chain ACL cached in *ACL Mgr* to check whether the requested topic partitions exist and whether the consumer has the *Describe* permission for the specified topic. If failed, returns the error response of unknown topic or partition to client. Otherwise, request handler exploits ACL manager to check whether the consumer has read access to the topic. For unauthorized topics, *TOPIC-_AUTHORIZATION_FAILED* response will be delivered to the consumer. Eventually, the request handler will employ *Replicas Mgr* to retrieve messages on specific topic partitions and return them. It should be noted that no matter whether the operation request on specified topics is successful or failed, a corresponding *TopicOper* instance will be generated and appended to the blockchain. The blockchain-enabled authorization process of *Produce* requests from producers is similar to that of *Fetch* requests from consumers.

Algorithm 1 Blockchain-enabled *Produce* Request Handler on BPS Broker

```
Input: msgList, producerID
Output: True or False
1: if CheckInvalid(producerID) then
2:     return False
3: end if
4: writeMsgList ← null
5: aclTopics ← CheckAclInChain(producerID, WRITE)
6: for each msg ∈ msgList do
7:     if msg.topic ∈ aclTopics then
8:         writeMsgList.Append(msg)
9:     else
10:        ReduceReputation(producerID)
11:        reputation ← GetReputation(producerID)
12:        if reputation<MIN_REPUATION then
13:            AddToBlackList(producerID)
14:        end if
15:        return False
16:    end if
17: end for
18: for each msg ∈ writeMsgList. do
19:     WriteRawMsgToLogStore(msg)
20:     WriteMetadataToChain(msg.topicPartition,
21:     msg.offset, msg.sizeInBytes)
22:     adminMgr.recordTopicOper(msg.topic,
23:     producerID, WRITE)
24: end for
25: AddReputation(producerID)
26: return True
```

4. BPS design and implementation

In this section, we demonstrate a detailed description for malicious behavior detection, data integrity verification, blockchain-enhanced access control, incentive mechanism, and batched-actions committing employed by BPS. To achieve the expected security goals with minimal performance overhead mentioned in Section 3, we implement a prototype system BPS based on Kafka and EoS blockchain. Taking publishing topic-classified messages by producers as an example, blockchain-enabled *Produce* request handler is leveraged by BPS broker, as shown in Fig. 5. Concretely, given the message list and producer ID provided by producers, after producer ID is authenticated, BPS broker will check all *Write*-authorized topics via on-chain ACL via Algorithm 4. Further, *Admin Mgr* in BPS broker filters out all valid messages according to corresponding *Write*-authorized topics. For each valid message: (1) *Log Mgr* writes the raw message to *Log Store*; (2) *Admin Mgr* writes message metadata to on-chain store via Algorithm 3; (3) *Admin Mgr* records the corresponding topic operation logs to blockchain through Algorithm 2.

To be specific, as shown in Algorithm 1, the validation of the producer is first checked at line 1. After that, a topic list is obtained from blockchain by calling Algorithm 4 at line 5. Then the group of messages to be produced is iterated, only messages whose topic with write permission will be produced. If a producer tries to produce a message whose topic without write permission, it will be rejected and incentive mechanism is called at line 10. The reputation of the producer will be reduced and it will be added to black list if the reputation is below a threshold, so this producer will no longer pass authorization at line 1. After messages with correct permissions are filtered, they are ready to

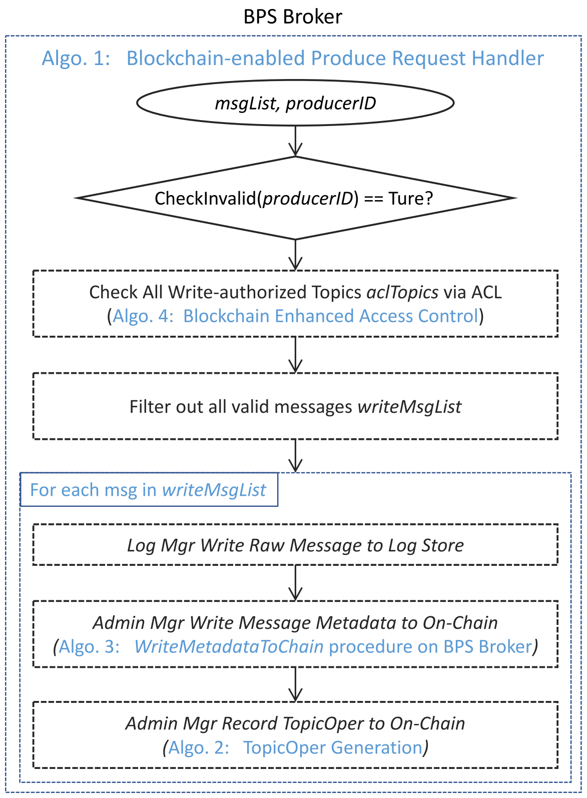

BPS Broker
Algo. 1: Blockchain-enabled Produce Request Handler
msgList, producerID
CheckInvalid(producerID) == Ture?
Check All Write-authorized Topics aclTopics via ACL (Algo. 4: Blockchain Enhanced Access Control)
Filter out all valid messages writeMsgList
For each msg in writeMsgList
Log Mgr Write Raw Message to Log Store
Admin Mgr Write Message Metadata to On-Chain (Algo. 3: WriteMetadataToChain procedure on BPS Broker)
Admin Mgr Record TopicOper to On-Chain (Algo. 2: TopicOper Generation)

Fig. 5. The overall relationship between Algo. 1, 2, 3, and 4 employed by BPS.

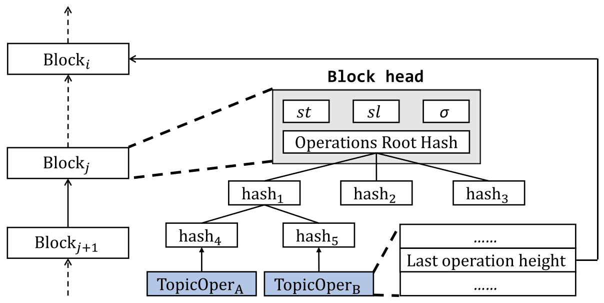


Fig. 6. Traceable topic operation in blockchain.

Table 3
TopicOper.

Field	Expression
operatorID	The ID of the client, i.e. operator
role	producer or consumer
topicID	the ID of the topic operated
operaName	WRITE, READ, DISCRIBE, DELETE
lastOpHeight	the block height of the *TopicOper*'s last detect operation
timestamp	The timestamp at the operator happened

be produced and written to local log files. After being written, the key metadata information is saved to blockchain by calling Algorithm 3 in order to achieve data integrity verification. Moreover, the operation logs of producers about topics are saved to the blockchain by calling Algorithm 2 at line 22, so we can achieve malicious behavior detection by analyzing the operation logs.

Algorithm 2 *TopicOper* Generation

```
Input: operatorID, role, topicID, operaName, timestamp
Output: TopicOper
1: if notexist(operatorID, role, topicID, operaName, timestamp then
2:     TopicOper ← WriteToChain(operatorID,
3:     role, topicID, operaName, timestamp)
4:     return TopicOper
5: else
6:     operaLog ← Find(topicID)
7:     lastOperh ← operaLog
8:     return TopicOper
9: end if
```

4.1. Malicious behavior detection

Considering malicious operations or misbehaviors monitoring, we propose an effective on-chain data structure *TopicOper* in Fig. 6 to store all topic operation logs of BPS into blockchain. The detailed fields and descriptions of *TopicOper* structure are shown in Table 3. When BPS broker handles the *Produce* or *Fetch* requests, Algorithm 2 is used to generate and record the corresponding operation logs. The field *lastOpHeight* refers to the block height of the last *TopicOper*. So, we can get the last operation logs without traveling over the blockchain again. A super administrator for BPS in the edge cloud can query all on-chain operation logs over the given topic by utilizing Algorithm 2 and then analyze the *TopicOper* logs to detect malicious behaviors taken by untrusted tenants. Thanks to the field *lastOpHeight*, we can efficiently obtain the entire historical topic operation logs without traversing the entire blockchain.

4.2. Data integrity verification

BPS employs a hybrid storage scheme where critical metadata is stored on-chain and raw messages data is persisted to log-based store. To ensure reliable integrity, Merkle Hash Trees (MHTs) on blockchain are employed to verify on-chain metadata while on-chain hashes are leveraged to authenticate off-chain data. More specifically, MHT is constructed through computation results on the basis of a one-way cryptographic hash operation (e.g., SHA256). An MHT in blockchain is constructed in a block by pairing transactions, then hashing until one single hash value (called Merkle Root) remains. In the binary tree of MHT, every leaf node containing one transaction can be verified via the corresponding path. By comparing the Merkle Root in each block, we can know whether the key metadata in the leaf nodes of MHT are tampered or not. In terms of off-chain data, when storing the raw messages to off-chain log store, a cryptographic hash of each message is generated by BPS and kept in blockchain, as shown in Algorithm 3 and line 20 of Algorithm 1. To verify the integrity of off-chain data, for each message in off-chain raw messages, we first get the corresponding on-chain hash and then use it to authenticate the message.

4.3. Blockchain-enhanced access control

When one BPS broker receives requests from producers or consumers, necessary authorizations are performed for topic-based access control. Original Kafka enforces access control which reply on ACL rules stored in centralized Zookeeper servers. Considering the security and immutability of blockchain, we record topic-related access control entries in the blockchain. Records are saved in tables where each record mainly consists of *producerID*, *topic*, *operation* fields. Take the message producing request from the producer as an example, as Algorithm 1, It needs to get access

Algorithm 3 *WriteMetadataToChain* procedure on BPS Broker

```
Input: topicPartition, offset, bytes
Output: True or False
1: procedure WriteMetadataToChain(topicPartition, offset, bytes)
2:     metadataTable ← GetTable()
3:     pKey ← metadataTable.AvailablePrimaryKey();
4:     if metadataTable.Insert(pKey, topicPartition, offset, bytes) then
5:         return True
6:     else
7:         return False
8:     end if
9: end procedure
```

control of topic before producing message. An authorize request *AuthRequest* $= \langle producerID, WRITE\rangle$ was sent to blockchain during the process. By calling Algorithm 4, the blockchain executes smart contract-enabled *find* function. In the smart contract, an index on the producerID is built and used to query the matching records according to producerID. After that, topics are filtered out according to *operationType* and a set of topics will be returned. Additionally, taking the high latency of querying ACL over smart contract into consideration, we cache hot ACL entries with expiration time in brokers.

4.4. Incentive mechanism

In order to enhance the security of BPS Broker as much as possible, we use EoS tokens to implement a heuristic incentive mechanism based on Routing to reward honest clients and punish malicious clients. Specifically, we create a blockchain account for each producer or consumer, and use EoS tokens to define a client's reputation which represents the credibility of a client. When clients are initialized, the BPS Broker assigns them a default reputation value. During the verification stage of request handler in BPS, when a producer attempts to write messages to unauthorized topics or when a consumer attempts to read messages from unauthorized topics, the corresponding reputation will be decreased. When the client's reputation is below a pre-defined threshold, it will be pushed into a blacklist by BPS broker. And when an honest client is processed successfully by the request handler of BPS broker, its reputation will increase appropriately. The details of such incentive mechanism are shown in Algorithm 1.

Algorithm 4 Blockchain Enhanced Access Control

```
Input: producerID, operation
Output: topicList
1: procedure CheckAclInChain(producerID, operation)
2:     indexTable ← IndexBy(byproducerid)
3:     records ← indexTable.Find(producerID)
4:     for each record ∈ records do
5:         if record.write is True then
6:             topicList.Append(record.topic)
7:         end if
8:     end for
9:     return topicList
10: end procedure
```

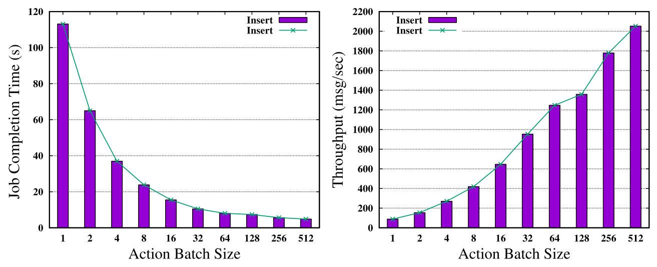


Fig. 7. The job completion time and throughput of recording *TopicOper* on-chain with different batch size.

4.5. Batched-actions committing

As blockchains ensure consistency and tamper resistance through a compute-intensive consensus protocol, the performance of blockchain systems is generally much worse than that of traditional database systems. In other words, the consensus algorithm is the main performance bottleneck of the blockchain system (EoS in our case). If each *TopicOper* is treated and stored as a transaction, the TPS (transactions per second) throughput of all existing blockchain is far from the requirement of high throughput in BPS. This is because the huge amounts of data produced by smart cities in a multi-tenant environment will generate large-scale concurrent requests. Such design choice makes the on-chain persistence of *TopicOper* dominate the steaming workflow of BPS, which induced performance overhead cannot be tolerated.

To tackle the above issue and adapt blockchain to BPS, we propose an asynchronous batched-actions committing scheme. Specifically, multiple *TopicOper* actions can be merged and stored into one transaction. Only when the number of cached *TopicOper* reaches a specified batch size or exceeds a predefined time window, a set of *TopicOper* instances will be committed to a transaction asynchronously through invoking corresponding smart contract program. The batch-based policy can dramatically improve *TopicOper* persistence throughput and utilization of blockchain storage resources in BPS. To determine an optimal batch size, we implement a smart contract that commits *TopicOper* actions in batches on the EoS Blockchain, and conduct the experiment to explore the relationship between batch size and TopicOper persistent throughput. The experimental results are shown in Fig. 7, and we found that *TopicOper*'s insert throughput increases linearly with increasing batch size. The throughput reaches its maximum when the batch size is 512. Therefore, in our case, BPS selects 512 as default batch size for *TopicOper* commit stage.

5. Evaluation

5.1. Security analysis

We present security profiling and discuss the effectiveness of various security mechanisms proposed by BPS. To give a more vivid explanation for the upcoming security proofs, we use two edge cloud-enabled smart city applications (smart medical and smart home IoT applications) as a scenario. The tenants of smart medical and home IoT applications are hospital and IoT device provider respectively. Specifically, for smart medical IoT applications, smart blood pressure meter and heart rate monitor are widely used to collect the patients' blood pressure and heart rate for behaviors and health monitoring. By publishing patients' health data to edge cloud, the required preprocessing and interference are performed to predict and assess the patients' health states. Note that these personal health datasets are significantly privacy-sensitive. For smart home IoT applications, smart air pollution monitors are generally used to collect indoor temperature, humidity, and pollution situation data. By publishing the indoor situation data to BPS brokers in edge cloud, the expected data mining tasks are performed to adjust the air conditioner/humidifier and assess the pollution index. These indoor situation data has relatively lower privacy requirements than the patients' health data. These two smart IoT applications share the same BPS services in edge cloud with different assigned topics. Meanwhile, the downstream Spark clusters employed by the above IoT apps fetch IoT data messages stream from BPS clusters for data mining and inference. We assume that the entire edge environment is untrusted.

Theorem 1. *In BPS, the topic-related operations such as Produce and Fetch messages can be traced efficiently to detect misbehaviors and illegal actions taken by malicious publishers or subscribers owned by untrusted tenants.*

Proof. We can observe that, for a given topic operation, we can efficiently get a full history of topic operation logs without traversing the entire blockchain. This is due to the trusted data structure *TopicOper* which realizes a topic operation chain for all clients whose topic operations are completely recorded on-chain. All queries and audits on *TopicOpers* are implemented based on trusted smart contract programs whose execution integrity is ensured by the blockchain consensus protocol. Furthermore, thanks to the tamper-resistant nature of blockchain, all *TopicOpers* cannot be modified by any unauthorized party. Thus, the query of any operations over topics can be fed back in an efficient and reliable manner. For instance, when a malicious tenant of smart home attempts to illegally publish junk messages to the assigned topics of the smart medical app and is refused by BPS brokers in edge cloud, the corresponding *TopicOper* has been generated and appended to the tamper-resistant blockchain. The attacking behavior of the smart home tenant will be audited and detected offline by the super administrator of edge cloud.

Theorem 2. *By decentralized design and TopicOper based audit, BPS can tolerate broker failure brought by DoS and other failures from traditional defense mechanisms realized in brokers under the threat model in Section 2.3.*

Proof. On the basis of characteristics of BPS described in Section 4.1, all topic-related operations must be stored in blockchain to be audited for anomaly behavior detection. Therefore, even if an attacker captures a BPS broker, all on-chain topic operations can be verified publicly by edge cloud super administrators and other BPS brokers. Thereby, the corrupted brokers will be detected quickly. On the other hand, in practice, BPS is deployed resilient edge cloud infrastructure, which makes it extremely difficult for an attacker to hinder BPS brokers from recording topic operations on-chain. Nevertheless, DoS attacks cannot be defended absolutely in an untrusted environment. We assume that a forceful attacker can interrupt accessing to BPS brokers via DoSing it. As all on-chain metadata (like ACLs and identity information of smart home and smart medical IoT applications) in BPS is in parallel with a decentralized fashion and each topic partition has multiple host-separated replicas, the brokers attacked by DoSing can be taken over by the others. But in traditional centralized pub/sub model like Kafka, all critical metadata is kept in centralized servers like Zookeeper. Once the centralized master servers are under attack, the result will be more terrible. Thus, compared with traditional centralized pub/sub models, BPS can ensure sustainable service under inevitable DoS attacks.

Theorem 3. *BPS can resist unauthorized topic operations and provide efficient data integrity verification with hybrid storage scheme.*

Proof. In terms of reliable authorization, BPS enforces fine-grained access control smart contracts over on-chain ACL (as described in Section 4.3), which means that ACL entries can be hardly tampered by inner attackers. Additionally, when it comes to integrity, security-sensitive on-chain metadata can be verified through Merkle Root in block header while large off-chain messages can be verified via corresponding on-chain hashes. Thus, we can detect data corruption through effective integrity verification (as introduced in Section 4.2) over the hybrid storage scheme. Particularly, we take the smart city IoT applications mentioned above as an example. For the blockchain-enabled hybrid storage scheme employed by BPS, patients' health data (such as blood pressure, heart rate) collected by smart medical devices, as well as indoor situation collected by smart home IoT devices (such as temperature, humidity, and pollution status) are stored into off-chain BPS broker *Log Store*, and the corresponding hashes are appended to the on-chain ledger in a tamper-proof and decentralized manner. Additionally, ACL and the mapping relationship between smart home/medical IoT apps and the assigned topics are stored in the blockchain. Naturally, blockchain-enhanced fine-grained access control and MHT-enabled integrity verification can provide reliable and secure BPS pub/sub services for the privacy-sensitive data/metadata generated by smart city applications.

Therefore, we believe that BPS can provide reliable and secure broker-based pub/sub streaming services in untrusted multi-tenant edge cloud.

5.2. Experiment setup

Hardware configuration: We conducted the performance experiment between BPS and plain Kafka on two servers. The DRAM size in each server is 128 GB with type of DDR4, whose speed is 2400 MHz. Each server is equipped with two E5-2687Wv4 CPU processors with 12 CPU cores and 3.00 GHz frequency. The L1 cache of each CPU core is 768 kB while the L2 cache is 3072 kB. 12 cores on one CPU processor share the same 30 MB L3 cache.

For performance comparison between BPS and Kafka, we concentrate on three import performance metrics including *auth time, job completion time (JCT)*, and *throughput (Tput)*. *Auth time* is defined as the time to verify related operation permissions. *Job completion time* is defined as how long it takes to complete *Produce* or *Fetch* requests when the specified number of messages is given. *Throughput* is defined as the number of messages produced or consumed per second. All experimental results are the average of five runs. In each run, produce and consume operations are executed under increasing message counts (2 million to 10 million) to compare the performance differences between BPS and Kafka.

In the next sections, we give a detailed comparison and analysis of the above-mentioned metrics of BPS and Kafka.

5.3. Auth time analysis

Fig. 8 shows the distribution of auth time when BPS/Kafka handles *Produce* or *Fetch* operations. We can find that with increasing scale of topic messages, the growth rate of *auth time* using BPS is gradually flattening. In the line chart in Fig. 8, the auth time gap between BPS and Kafka represents the authentication overhead introduced by on-chain fine-grained ACL and incentive mechanism in BPS. Thanks to the optimization of caching hot ACL entries with an expiration time in brokers, the authentication overhead gap gradually shrinks convergently

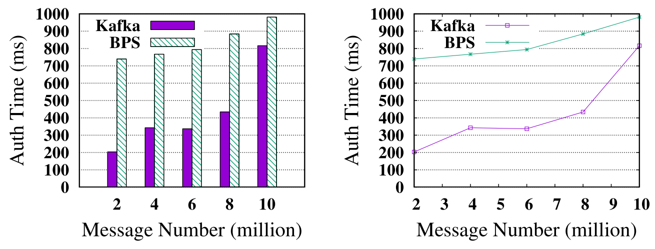


Fig. 8. Comparison of auth time between BPS and Kafka with increasing message number.

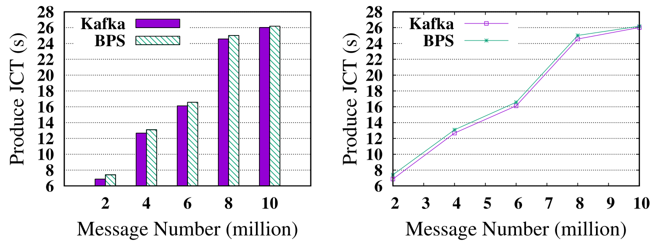


Fig. 9. Comparison of producer's job completion time between BPS and Kafka with increasing message number.

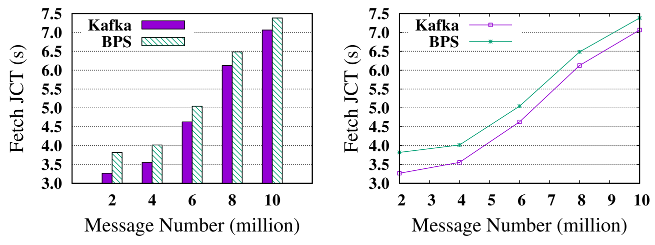


Fig. 10. Comparison of consumer's job completion time between BPS and Kafka with increasing message number.

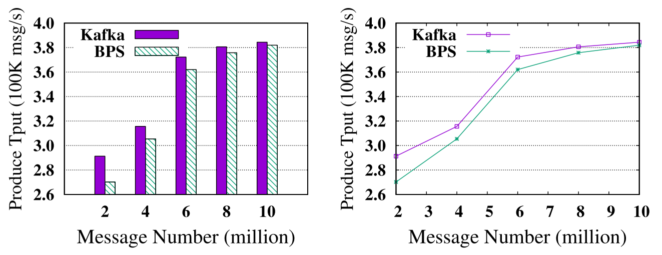


Fig. 11. Comparison of producer's throughput between BPS and Kafka with increasing message number.

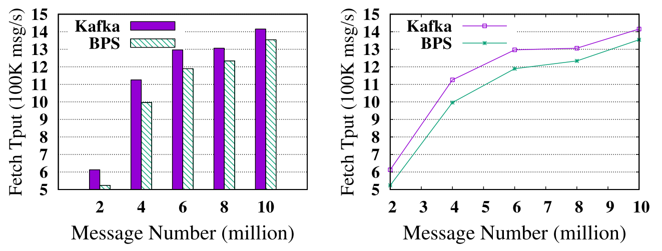


Fig. 12. Comparison of consumer's throughput between BPS and Kafka with increasing message number.

with increasing message scale. The adoption of batched-actions committing is another decisive factor which increases the gains made each time the blockchain is queried or appended. Compared with Kafka's auth time growth of up to nearly 50% with increasing message scale, BPS can maintain a relatively stable growth rate of only about 10%. When the message number reaches 10 million, the auth time using BPS is only up to 10% more than Kafka.

The above data analysis shows that our BPS can ensure reliable authentication and verification while keeping a low and stable overhead under large-scale message processing. It is worth noting that it is not possible that BPS will exceed Kafka. Because of the introduction of blockchain verification operations and incentive mechanisms, the additional overhead is inevitable. But the good news is that as the message data increases, the gap between the two is decreasing and converging.

5.4. Job completion time analysis

As shown in Figs. 9 and 10, the comparison of both the producer's and the consumer's job completion time between BPS and Kafka is also illustrated *Fetch* operation from the consumer in Fig. 10, where there is a JCT gap close to 0.6 s when the number of messages is 2 million. However, when the message scale increases to 10 million, the gap between BPS and Kafka is reduced to a maximum of about 0.3 s. Similar conclusions are more easily confirmed in the polyline charts of figures 9 and 10. Both *Produce* and *Fetch* JCT show a significant increase as the message scale increases. What we need to pay attention to is that in the two polyline charts, the corresponding polylines of BPS and Kafka gradually converge as the message scale increases. In the polyline graph of Fig. 9, it can even be observed that the last two polylines almost intersect at the same point-around 26 s. All in all, BPS can achieve a stable and tolerable authentication overhead, with an extra maximum proportion of 7% and minimum proportion of 0.19% of Kafka.

We find that the Producer's JCT is generally higher than the Consumer's, which is due to more frequent and strict security certificate as well as the different batch size between *Produce* and *Fetch* operations. First, compared to consumers inside the edge cloud (usually stream processing systems), BPS external producers (e.g. IoT applications) are owned by various untrusted tenants. Therefore, they require more frequent and strict security authentication, such as access control and incentive mechanism. Second, the default batch size for the *Produce* operation adopted by BPS and Kafka is much smaller than that of the *Fetch* operation. We take full advantage of high EoS transaction throughput and cache the hot ACL entries in local broker nodes to reduce the extra verification overhead. As the message scale increases, the additional security certification overhead gap between BPS and Kafka gradually shrinks.

5.5. Throughput analysis

Figs. 11 and 12 illustrate the comparison of the producer and consumer's *throughput* when using BPS and Kafka under increasing message numbers. We can find that BPS can achieve almost the same throughput as Kafka with little overhead. To be more specific, the throughput of BPS can achieve 93% of Kafka when the message number is 2 million, and can achieve 99.4% of Kafka when message number reaches 10 million. Under the same increasing number of messages, the produce operation still performs better than the fetch operation in terms of throughput. From the polyline chart, we can see that the change trends of the two are almost the same, which shows that the additional overhead of BPS is relatively stable, has good robustness. Also, the throughput growth trend of producers tends to ease after the message number of 8 million. We will give a specific analysis as follows.

First of all, we can clearly find that the throughput of the fetch operation is significantly larger than the produce operation, which is due to its pull-based model. In our scheme, we designed to enable broker nodes to return messages in batches. This mode makes the throughput relatively large. In addition, in our scheme we designed and introduced blockchain-based access control, on-chain TopicOper-based monitoring, data integrity verification and the incentive mechanism, which will inevitably become a non-negligible overhead when the amount of messages is small. But as the amount of messages increases, thanks to the batched-actions committing we proposed and the high transaction throughput of EoS blockchain, the throughput gap between BPS and Kafka has been significantly reduced.

6. Conclusion & future work

To accommodate the demand for confidentiality and reliability in untrusted multi-tenant edge cloud, this paper presents *BPS*, a blockchain-enhanced, reliable, and efficient architecture for general topic-based broker-enabled pub/sub communication model. We first analyze and identify the safety bottleneck including unauthorized access or tampering with critical privacy-sensitive metadata induced by the centralized schemes in traditional pub/sub systems. Inspired by the prevalent blockchain enhanced paradigm, we attempt to employ the attractive on-chain properties of decentralization, traceability, and immutability to redesign the security mechanism of the broker-enabled pub/sub system model. Specifically, BPS gives a new trusted data structure *TopicOper* which is recorded in blockchain for forward traceability and history misbehaviors detection. A smart contract-enabled fine-grained access control via on-chain ACL is leveraged to ensure reliable and tamper-resistant authorization for topic operations, while a reputation-based incentive mechanism is proposed to punish potential malicious clients owned by untrusted tenants. To scale up BPS brokers, we provide a hybrid storage scheme with on-chain key metadata and off-chain raw topic messages. Regarding data integrity, Merkle Hash Tree (MHT) is exploited to verify on-chain metadata while on-chain hashes are utilized to authenticate off-chain raw data. Furthermore, batched-actions committing and caching policy are employed to minimize performance overhead. We implement BPS prototype based on Kafka and EoS blockchain. The security analysis and extensive experiments compared with original state-of-the-art Kafka show that BPS is suitable in practice.

In our future work, we may extend BPS to support larger-scale and more complex deployments in edge cloud as well as verify and improve the scalability of BPS. Considering the differentiated characteristics and security needs of various upstream edge applications, we will further explore deadline-aware and security-adaptive pub/sub model based on BPS to bridge the gap between edge application-level (security & performance) goals and service-level optimization.

CRediT authorship contribution statement

Bobo Huang: Resources, Project administration, Writing - original draft, Supervision. **Rui Zhang:** Conceptualization, Methodology, Writing - original draft, Software. **Zhihui Lu:** Data curation, Writing - review & editing, Supervision, Funding acquisition. **Yiming Zhang:** Investigation, Formal analysis, Writing - original draft, Validation. **Jie Wu:** Writing - review & editing, Supervision. **Lu Zhan:** Data curation, Visualization, Writing - review & editing. **Patrick C.K. Hung:** Visualization, Writing - review & editing.

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.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments and suggestions. The work of this paper is supported by National Key Research and Development Program of China (2019YFB1405000), National Natural Science Foundation of China under Grant (No. 61873309, No. 61572137, and No. 61728202), and Shanghai Science and Technology Innovation Action Plan Project under Grant (No. 19510710500, No.18510760200, and No. 18510732000).

References

[1] Amazon, Amazon serverless data processing, 2019, https://cloud.google.com/pubsub/.
[2] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S.W. Cocco, J. Yellick, Hyperledger fabric: A distributed operating system for permissioned blockchains, in: Proceedings of the Thirteenth EuroSys Conference, in: EuroSys '18, Association for Computing Machinery, New York, NY, USA, 2018, http://dx.doi.org/10.1145/3190508.3190538.
[3] P. Anusree, S. Sreedhar, A security framework for brokerless publish subscribe system using identity based signcryption, in: 2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015], IEEE, 2015, pp. 1–5.
[4] Apache, Pulsar: distributed pub-sub messaging system, 2019, https://pulsar.apache.org/.
[5] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J.A. Kroll, E.W. Felten, Mixcoin: Anonymity for bitcoin with accountable mixes, in: International Conference on Financial Cryptography and Data Security, Springer, 2014, pp. 486–504.
[6] H. Cao, R. Li, W. Tian, Z. Xu, W. Xiao, Blockchain-based accountability for multi-party oblivious ram, J. Parallel Distrib. Comput. 137 (2020) 224–237.
[7] J. Chen, S. Yao, Q. Yuan, K. He, S. Ji, R. Du, Certchain: Public and efficient certificate audit based on blockchain for tls connections, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, IEEE, 2018, pp. 2060–2068.
[8] X. Chen, Z. Zhao, C. Wu, M. Bennis, H. Liu, Y. Ji, H. Zhang, Multi-tenant cross-slice resource orchestration: A deep reinforcement learning approach, IEEE J. Sel. Areas Commun. 37 (10) (2019) 2377–2392.
[9] M. Conti, M. Hassan, C. Lal, Blockauth: Blockchain based distributed producer authentication in icn, Comput. Netw. 164 (2019) 106888.
[10] H.-N. Dai, Z. Zheng, Y. Zhang, Blockchain for internet of things: A survey, IEEE Internet Things J. 6 (5) (2019) 8076–8094.
[11] DanielLarimer, Eosio, blockchain software architecture, 2020, https://eos.io/.
[12] J. Daubert, M. Fischer, T. Grube, S. Schiffner, P. Kikiras, M. Mühlhäuser, Anonpubsub: Anonymous publish-subscribe overlays, Comput. Commun. 76 (2016) 42–53.
[13] P. Dobbelaere, K.S. Esmaili, Kafka versus rabbitmq: A comparative study of two industry reference publish/subscribe implementations: Industry Paper, in: Proceedings of the 11th ACM International Conference on Distributed and Event-Based Systems, 2017, pp. 227–238.
[14] A. Dorri, S.S. Kanhere, R. Jurdak, P. Gauravaram, Lsb: A lightweight scalable blockchain for iot security and anonymity, J. Parallel Distrib. Comput. 134 (2019) 180–197.
[15] M. El-Hindi, M. Heyden, C. Binnig, R. Ramamurthy, A. Arasu, D. Kossmann, Blockchaindb-towards a shared database on blockchains, in: Proceedings of the 2019 International Conference on Management of Data, 2019, pp. 1905–1908.
[16] H.M. Elkamchouchi, E. Elkheir, Y. Abouelseoud, A pairing-free identity based tripartite signcryption scheme, Int. J. Cryptogr. Inf. Secur. (IJCIS) 3 (4) (2013).
[17] I. Eyal, A.E. Gencer, E.G. Sirer, R. Van Renesse, Bitcoin-ng: A scalable blockchain protocol, in: 13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16), 2016, pp. 45–59.
[18] Google, Google cloud pub/sub, 2019, https://cloud.google.com/pubsub/.
[19] A. Hari, M. Kodialam, T. Lakshman, ACCEL: Accelerating the bitcoin blockchain for high-throughput, low-latency applications, in: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, IEEE, 2019, pp. 2368–2376.

[20] B. Huang, L. Jin, Z. Lu, X. Zhou, J. Wu, Q. Tang, P.C. Hung, Bor: Toward high-performance permissioned blockchain in RDMA-enabled network, IEEE Trans. Serv. Comput. (2019).
[21] IBM, IBM cloud functions, 2019, https://www.ibm.com/cloud/functions.
[22] M. Ion, G. Russello, B. Crispo, Design and implementation of a confidentiality and access control solution for publish/subscribe systems, Comput. Netw. 56 (7) (2012) 2014–2037.
[23] M. Javed, X. Lu, D.K. Panda, Characterization of big data stream processing pipeline: a case study using Flink and Kafka, in: Proceedings of the Fourth IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, 2017, pp. 1–10.
[24] J. Koshy, Kafka ecosystem at linkedin, 2019, https://engineering.linkedin.com/blog/2016/04/kafka-ecosystem-at-linkedin.
[25] J. Kreps, N. Narkhede, J. Rao, et al., Kafka: A distributed messaging system for log processing, in: Proceedings of the NetDB, Vol. 11, 2011, pp. 1–7.
[26] M. Li, J. Weng, A. Yang, W. Lu, Y. Zhang, L. Hou, J.-N. Liu, Y. Xiang, R.H. Deng, Crowdbc: A blockchain-based decentralized framework for crowdsourcing, IEEE Trans. Parallel Distrib. Syst. 30 (6) (2018) 1251–1266.
[27] Z. Lu, N. Wang, J. Wu, M. Qiu, Iotdem: An iot big data-oriented mapreduce performance prediction extended model in multiple edge clouds, J. Parallel Distrib. Comput. 118 (2018) 316–327.
[28] P. Lv, L. Wang, H. Zhu, W. Deng, L. Gu, An iot-oriented privacy-preserving publish/subscribe model over blockchains, IEEE Access 7 (2019) 41309–41314.
[29] B. Maithily, Y. Swathi, Securing broker-less publish/subscribe system using fuzzy identity-based encryption, Int. J. Comput. Sci. Inf. Technol. 6 (3) (2015) 2823–2826.
[30] V.D. Malpure, P. Deshmukh, Provide security for broker-less content based publish system using pairing based cryptography, Int. J. Eng. Develop. Res. 4 (2) (2016) 1932–1938.
[31] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, Technical Report, Manubot, 2019.
[32] S. Nathan, C. Govindarajan, A. Saraf, M. Sethi, P. Jayachandran, Blockchain meets database: design and implementation of a blockchain relational database, Proc. VLDB Endow. 12 (11) (2019) 1539–1552.
[33] Pivotal, Rabbitmq, 2019, https://www.rabbitmq.com/.
[34] S. Shitole, A. Gujar, Securing broker-less publisher/subscriber systems using cryptographic technique, in: 2016 International Conference on Computing Communication Control and Automation (ICCUBEA), IEEE, 2016, pp. 1–6.
[35] S.-F. Sun, M.H. Au, J.K. Liu, T.H. Yuen, Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero, in: European Symposium on Research in Computer Security, Springer, 2017, pp. 456–474.
[36] M.A. Tariq, B. Koldehofe, A. Altaweel, K. Rothermel, Providing basic security mechanisms in broker-less publish/subscribe systems, in: Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems, 2010, pp. 38–49.
[37] M.A. Tariq, B. Koldehofe, K. Rothermel, Securing broker-less publish/subscribe systems using identity-based encryption, IEEE Trans. Parallel Distrib. Syst. 25 (2) (2013) 518–528.
[38] A.V. Terkhedkar, M.A. Shah, Providing security mechanisms in broker-less publish/subscribe systems using hierarchical identity based encryption, in: 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), IEEE, 2016, pp. 641–645.
[39] S. Wang, T.T.A. Dinh, Q. Lin, Z. Xie, M. Zhang, Q. Cai, G. Chen, B.C. Ooi, P. Ruan, Forkbase: An efficient storage engine for blockchain and forkable applications, Proc. VLDB Endow. 11 (10) (2018) 1137–1150.
[40] G. Wood, et al., Ethereum: A secure decentralised generalised transaction ledger, in: Ethereum Project Yellow Paper, Vol. 151, 2014, pp. 1–32.
[41] S. Xie, Z. Zheng, W. Chen, J. Wu, H.-N. Dai, M. Imran, Blockchain for cloud exchange: A survey, Comput. Electr. Eng. 81 (2020) 106526.
[42] C. Xu, K. Wang, P. Li, S. Guo, J. Luo, B. Ye, M. Guo, Making big data open in edges: A resource-efficient blockchain-based approach, IEEE Trans. Parallel Distrib. Syst. 30 (4) (2018) 870–882.
[43] K. Yang, K. Zhang, X. Jia, M.A. Hasan, X.S. Shen, Privacy-preserving attribute-keyword based data publish-subscribe service on cloud platforms, Inform. Sci. 387 (2017) 116–131.
[44] ZeroMQ, Zeromq, an open-source universal messaging library, 2019, http://zeromq.org/.
[45] C. Zhang, C. Xu, J. Xu, Y. Tang, B. Choi, Gemˆ 2-tree: A gas-efficient structure for authenticated range queries in blockchain, in: 2019 IEEE 35th International Conference on Data Engineering (ICDE), IEEE, 2019, pp. 842–853.
[46] Y. Zhao, Y. Li, Q. Mu, B. Yang, Y. Yu, Secure pub-sub: Blockchain-based fair payment with reputation for reliable cyber physical systems, IEEE Access 6 (2018) 12295–12303.
[47] Y. Zhao, Y. Liu, A. Tian, Y. Yu, X. Du, Blockchain based privacy-preserving software updates with proof-of-delivery for internet of things, J. Parallel Distrib. Comput. 132 (2019) 141–149.
[48] P. Zheng, Z. Zheng, H.-n. Dai, Xblock-eth: extracting and exploring blockchain data from etherem, 2019, arXiv preprint arXiv:1911.00169.

Bobo Huang is a PH.D. student at School of Computer Science, Fudan University. His research interests are cloud computing, distributed system, data center networks, big data system, Blockchain distributed system, network functions virtualization, network management driven by Big Data.

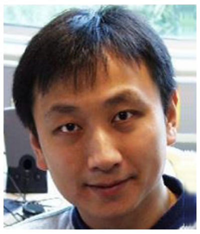
Rui Zhang is a master student at School of Computer Science, Fudan University. His research interests are computer network, cloud computing, Blockchain distributed system, IoT and big data analysis system.

Zhihui Lu is a Professor in School of science degree IEEE and committee. Computer Science, Fudan University. He received a Ph.D computer from Fudan University in 2004, and he is a member of the China computer federation's service computing specialized committee. His research interests are cloud computing and service computing technology, big data architecture, edge computing, and Blockchain distributed system.

Yiming Zhang is a master student at School of Computer Science, Fudan University. His research interests are networking, distributed system, data center networks, cloud computing, network architecture.

Jie Wu is a Professor at School of Computer Science, Fudan University. His research interests are Internet technology, big data architecture, edge computing, cloud computing, Blockchain distributed system, he received a Ph.D computer science degree from Fudan University in 2008.

Lu Zhan is an assistant professor at School of Economics, Fudan University. Her research interests are Economics Big data, Blockchain in Economics.



Patrick C.K. Hung is a Professor at the Faculty of Business and Information Technology in University of Ontario Institute of Technology. Patrick has been working with Boeing Research and Technology in Seattle, Washington on aviation services-related research projects. His research interests include services computing, cloud computing, big data, edge computing.