IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 20, 2021, accepted June 18, 2021, date of publication June 28, 2021, date of current version July 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3092732

Secure-Stor: A Novel Hybrid Storage System Architecture to Enhance Security and Performance in Edge Computing

MAIS NIJIM[1], (Senior Member, IEEE), AND HISHAM ALBATAINEH[2]
[1]Department of Electrical Engineering and Computer Science, Texas A&M University-Kingsville, Kingsville, TX 78363, USA
[2]Department of Physics and Geosciences, Texas A&M University-Kingsville, Kingsville, TX 78363, USA

Corresponding author: Mais Nijim (mais.nijim@tamuk.edu)

This work was supported by the Department of Homeland Security under Grant 21STSLA00011-01-00.

This work did not involve human subject or animals in its research.

ABSTRACT The Internet of Things (IoTs) is attracting the attention of scientists worldwide. With its explosive growth along with applications that require real-time computing power, a new technology called edge computing has emerged. As a result, edge computing has changed the way data are processed and handled back and forth for millions of devices worldwide, such as autonomous vehicles and electric cars. The confinement of cloud computing technology, such as a content delivery network (CDN), contributed significantly to edge computing development. Currently, this technology can meet the demands of ever-growing mobile devices and the IoTs. This paper describes a novel secure framework consisting of a hybrid storage architecture consisting of CDN, edge computing, and centralized storage. Centralized storage consists of multilevel storage systems that comprise solid-state drivers (SSDs) and hard disk drivers (HDDs), which provide optimal data storage solution for a wide variety of real-time data processing applications. Transforming the data back and forth between SSDs and HDDs is crucial for achieving high performance while meeting the edge device request deadline. Additionally, a new dynamic solid-state disk partitioning mechanism is introduced to optimize security for the proposed framework among hard disk drives. A partition from the solid-state disks to hard disk drives is assigned based on hard disk drive workloads.

INDEX TERMS Hybrid storage systems, machine learning, satisfied ratio, security level.

I. INTRODUCTION

Cloud computing infrastructure generally provides various services, such as networking, storage, and computation for individual organizations, reducing the burden of edge devices. The cloud servers of cloud computing facilities have enough space for storage and computing capacity, which is beneficial for many users. Being far away from local devices, cloud computing does not support distributed Internet of Things (IoTs) environments to a maximum extent. IoTs applications need to support location awareness, mobility, a response in real-time and context awareness, restricting the freedom to use cloud computing far away from user machines. One of the cloud computing's essential functions is that it manages a tremendous amount of raw data, which might be the bottleneck of present networks or laid infrastructure primarily due to queuing delays. To address these issues, edge computing has been developed.

Data are stored in more places, in a different format in a larger quantity than ever before. From autonomous cars, oil rigs, and factories, there is a need for real-time data processing. One solution for all such problems is edge computing. Currently, public clouds, such as content delivery networks (CDNs) [1], are widely used due to their ability to compute and store data on a large scale. CDN is the transparent backbone of the delivery of content on the Internet. A CDN stores a cached version of its content in different geographic locations to reduce the distance between the website's server and visitors. The CDN contains many caching servers responsible for providing visitors within its vicinity with information and placing content in a variety of locations at once, providing customers with superior coverage. A CDN is a geographically based data center and can communicate in its corresponding geographical region with users. Its primary objective is to

The associate editor coordinating the review of this manuscript and approving it for publication was Tai-Hoon Kim.

reduce time by taking the information closer to the website visitor. Each CDN comprises several caching servers. Each caching server is responsible for the delivery and storage of cached files. Its primary purpose is to speed up load times for websites. Each CDN caching server usually keeps multiple hard drives and high quantities of RAM resources. In CDN caching servers, cached files are stored on hard-disk and solid-state drives.

Limitations on the size of internet pipelines and light speed create some problems while moving the cloud's stored data. With edge computing [2], data are processed at the source level, which provide real-time insights, and it is not sent to the cloud. This creates an environment that acts as a public cloud. In recent years, the rapid advances in smart mobile apps and the IoTs [3] have significantly enabled edge computing [4] to advance. In edge computing, generous support is given for IoTs devices to perform difficult tasks in the best way. It avoids large threats occurring in the factor of security platforms and applications. Edge computing is simply described as the processing and transmission of data from devices distributed worldwide. The rapid development of devices associated with networking technologies, IoTs, and their applications continues to drive edge computing frameworks. The fast growth in networking technologies, such as wireless networking technology, accelerates development and enables real-time applications, such as self-driving vehicles, artificial intelligence video processing, and analytics. Edge computing works on instant data, real-time data produced by the sensor, and real-time users. Alternatively, cloud computing is a technology that runs on big data.

Edge computing provides location consciousness, sufficient bandwidth, privacy, and real-time and low-cost infrastructure to keep up with emerging smart city applications. These advantages over the cloud have contributed to the growth of edge computing. Edge computing's market size is increasing every day. The accomplishments of the IoTs and lush cloud services helped establish a new generation of edge computing. Data processing takes place at the network edge instead of processing the data entirely in the cloud. It could address issues such as cost of bandwidth, latency, limited battery life, privacy, and security. Moving computing activities to the cloud network is considered an efficient method to handle data, as the cloud has more processing capacity than computers at the edge network.

Moreover, while processing data speeds have risen exponentially, no massive increase in the networks' bandwidth has occurred that carry data to the cloud. Simultaneously, the system is becoming the bottleneck of a cloud with edge devices producing more data. For example, an autonomous vehicle camera can capture a vast amount of raw data that the system needs to process in real time to generate better driving results without latency. If there is no edge server, the data need to be processed in the cloud, and the response time will be longer where the response efficiency is affected. Autonomous vehicles are one area that would further strain network reliability and bandwidth. Processing data at the edge is more productive and efficient; it produces shorter responses that will lower pressure on the network. Interconnectivity, which has increased tremendously, provides access for more applications with improved edge computing. A new IoTs and specific business use cases of the industry are used. Edge computing-based infrastructure is approved to be one of the best storage servers and has immense growth in the future.

There are several factors [5] that determine cloud computing that are not adequate for use alone. The first important factor is latency. Edge computing can provide latency in milliseconds by sending data to the edge server, and a high response time can be obtained. In the cloud computing model, applications need to send data to the data center and then attain a response, which significantly increases the latency period of the system. The second factor is the high throughput: Throughput is available to the user from the edge, served via locally generated or cached content. In the proposed framework, data will be cashed in an array of solid-state disks. The third factor is data reduction when running edge-based computing applications, such as data analytics. Operators and application providers may significantly reduce the amount of data submitted upstream, often used to achieve cost savings. The fourth factor is data availability, where there are currently an increasing number of cloud-based Internet services; the use of such services has become an essential part of our everyday lives. The fifth important factor is that security cloud service providers can also secure their networks against attacks from customer premise equipment or user equipment using edge security. There is a concern with high-speed connections in the number of places not always linked to the Internet. During times of disrupted or missing connectivity, an edge cloud can provide services that provide disaster resilience.

Hybrid storage systems provide a reduced cost for data centers without affecting the requested response times. In the proposed hybrid storage systems, two levels in the centralized location are used, which are consistent with the solid-state disks in the upper level and hard disk drives in the lower level. The most frequently used data is placed on the upper level. The data requests are first checked at the upper-level storage device, which is the SSDs. If the data are not found in the upper level, the lower-level device, which is the HDD, is checked. These steps are constantly repeated until the data request is found.

Security services for the above architecture are essential to protect and secure residing data from talented intruders [6]. In addition, many edge server requests must be finished before their absolute deadline [7]. Through this thesis work, a hybrid storage framework that integrates several storage devices is proposed, i.e., CDN, edge server connected with the CDN, and an array of solid-state disks. In the central location, a hybrid storage system architecture consisting of a combination of solid-state disks and a variety of hard disk drives are used. The proposed framework uses security control protocols that can be adjusted to encounter security change necessities and workload environments, providing

a high security level for all edge device requests. A prominent feature of the proposed framework is the solid-state disk partitioning mechanism, which helps to increase the security level of the proposed framework while maintaining the desired deadline.

As current edge servers do not include hybrid storage to improve security level (SL) quality, the next-generation parallel system server is intended to be highly scalable to maintain differing SLs under different conditions against experienced intruders, such as modification of the SL or a changing workload. There is an even greater need for security optimization mechanisms in various real-time applications, as processing and sensitive data content require specific protections against unauthorized access.

This paper proposes a hybrid storage framework that integrates several storage devices, i.e., CDNs, edge servers that is connected with CDNs, and an array of solid-state disks. In the central location, we use a hybrid storage system architecture consisting of a combination of an array of solid-state disks and an array of hard disk drives. The proposed framework uses security control protocols that can be adjusted to encounter security change necessities and workload environments; therefore, providing a high level of security for all edge device requests. A prominent feature of the proposed framework is the use of the solid-state disk partitioning mechanism that helps in incrementing the level of security of the proposed framework while maintaining the desired deadline.

The paper is organized as follows. Section 2 describes the background and the problem statement. In section 3, we summarize the literature review. Section 4 describes the proposed architecture and the data management plan. In section 5, we describe the proposed algorithm. Section 6 analyses the experimental results. Lastly, section 7 concludes the paper with future recommendations.

II. BACKGROUND AND PROBLEM STATEMENT

A. BACKGROUND

1) SECURITY IN EDGE COMPUTING

To support different IoTs applications and extend cloud computing to the edge of a network, edge computing is considered a groundbreaking technology. Although edge computing is beneficial, they might still have numerous privacy and security issues. The basic definition and functionality of edge computing are compared to cloud computing. The potential privacy and security threats to propose several performance requirements and security requirements are analyzed. Much advanced secure data analytics in edge computing is reviewed. A variety of problems related to safe data analytics in edge computing are present. There is a demand and need for real-time response in edge computing from some IoTs applications. The question of how to manage efficiency and security is still unanswered. A very significant number of works require user devices to run certain complex operations that incur enormous costs for computing. Complex computing influences performance specifically for consumer devices whose resources are constrained. Therefore, in many realistic circumstances, making a tradeoff between efficiency and security becomes essential.

Neither edge nodes nor cloud servers can be trusted entirely in edge computing, making it challenging to ensure processing, analytics, and computing data correctness. There is no practical approach for testing data accuracy when using edge nodes to perform data analytics. In outsourced data analysis, the correctness of the calculation remains of primary importance. If no security solution is available to ensure accuracy, end users would not be able to use edge computing technology. To classify end-user data sensitivity and manage those data files, edge computing becomes the main problem to achieve robust and stable data analytics, e.g., geographic information and health status, while some are not, e.g., climate-related data and social events.

2) MACHINE LEARNING TECHNIQUES IN EDGE COMPUTING

Due to a lack of training data and computational power, machine learning models are not commonly used [8]. However, machine learning can now be realized due to improved computing capability and extensive data availability to train machine learning algorithms. The caching of most known content close to user devices, i.e., at edge nodes, is the primary approach for enhancing customer service efficiency on future generation networks. However, it is challenging to accurately forecast the popularity of the content and determine which content needs to be stored in the cache of the server. Machine learning algorithms are used to predict and learn the popularity of content. They have achieved high prediction precision with developments in increased computing power and big data. In the cloud data center, machine learning models [9] are trained and used to make a successful decision on the cache. Then, to store the typical content proactively, the decision is sent to each server.

Edge nodes offer access to a community of close people who have common interests in content. Caching common data on edge nodes while reducing the core networks' load contributes to lower latencies [10]. To measure the success of content from a global viewpoint, social and temporal characteristics of content, such as likes and number of people who watched, are applied. The dominant dynamic features of edge networks include user versatility, expectations, and popularity of content. Machine learning approaches can be used in edge networks to predict content popularity based on end-user expectations, replacement strategies, related content interests, and to optimize cache replacement. They have given a collection of network state predictions and limitations. These machine learning applications can assist in defining appropriate content for an edge network. Modern mobile apps require low latency and have support for mobility, high bandwidth, and energy efficiency from backend data stores that are usually hosted in content delivery networks (CDNs) and cloud data centers.

Cache refers to in-network storage that includes the content of the request that is frequently accessed. In the last decade,

edge networks have addressed high content latency problems while lowering the burden on backhaul networks with the aid of network caching. For tasks involving clustering, grouping, prediction, and ML techniques are helpful [11]. Machine learning (ML) offers computer systems the opportunity to learn from experience without the need for explicit programming. In this case, the experience is the dataset on which the algorithms used to train themselves. With time, the models in the dataset will discover the underlying trends and patterns. These models can make accurate predictions after good learning and, thus, provide predictive analytics.

3) MACHINE LEARNING MODES

There are two types of machine learning styles: supervised and unsupervised. Supervised learning allows one to collect data or produce output data from experience. It helps to optimize performance criteria using experience. In contrast, unsupervised learning discovers all sorts of unidentified trends in the information, and unsupervised techniques help identify functionality that can be helpful for classification. Therefore, supervised learning aims to learn a model that best predicts the relation between the observable output and input content. In the classification area, supervised learning usually occurs when the input is mapped to the output labels. The objective of both classification and regression is to identify specific relationships or structures in input data to produce correct output data.

B. PROBLEM STATEMENT

In the public and private sectors, IoTs technologies are becoming pervasive and are currently an integral part of our everyday lives. The benefits provided by these innovations are often associated with severe security concerns that are often not adequately managed or even ignored. The IoTs threat environment is vast and diverse and requires various technologies for software and hardware. IoTs devices such as cameras, health monitors, and smart bulbs are actively adopted by consumers, with figures expected to grow to billions. However, such devices are often easily attacked or used to launch attacks on a wide scale and increasing frequency. IoTs devices are exposed to various security threats due to the lack of adequate protocol security services and limitations or incorrect configuration of the products and services being deployed. The cloud computing paradigm does not meet demands due to centralized processing and is far from local computers. To store and process data at the edge of networks, edge computing was then implemented and is closer to data sources than cloud computing, making it powerful and location conscious. Unexpectedly, as applied to data analytics, edge computing presents privacy and security problems.

While edge servers offer several advantages, they also face several threats including privacy and security issues. Since edge computing is a cloud computing extension, it has some cloud computing issues with security. Due to its distinctive characteristics, such as low latency and geographic spread, edge computing often has privacy and security issues. The implementation of security frameworks is indispensable for achieving safe data analytics. Unfortunately, the cloud system's standard protection frameworks are not appropriate for the edge system due to limited edge device resources. Therefore, it is necessary to build security solutions for edge computing to support efficient and reliable IoTs-based edge applications.

Data protection requires ensuring that unauthorized snoopers do not understand shared data between endpoints. Using three steps, whether encoded, plaintext, or encrypted, the confidentiality of exchanged data can be determined. All communication channels are analyzed between the device and user app, device and cloud server, user app and cloud server of a given IoTs device. Therefore, various encryption algorithms are used to provide maximum protection and complete the IoTs request within the given deadline.

There are difficulties in integrating edge computing with the IoTs and with the advantages that edge computing can provide. The efficient handling of edge infrastructure and assigning resources available to IoTs devices is one potential problem. At any time, IoTs devices can request various services, and the computation and storage capacity of each edge server node is limited. When the edge server is assigned to provide services, various criteria need to be addressed. Another issue is that the edge server can be optically mapped to IoTs devices to meet IoTs application requirements. Furthermore, protection and privacy problems also remain.

In each computing paradigm, data protection and privacy conservation are important issues, especially in edge computing. For an outsourcing situation, processing situations and data storage become more complicated. Summarizing a few open research questions concerning privacy and data protection must be addressed before edge computing is deployed. To achieve distributed and lightweight and data encryption systems, edge computing's accessible properties, such as resource constraints, parallel computing, the coexistence of multiple trust domains, and massive data processing, should be thoroughly considered in design encryption processes.

Users create massive data at the edge network, and this created data are measured in part or entirely on edge machines. In the dynamic data updating process, most of the existing privacy management methods do not have a dynamic update feature, so the security of fine-grained data and privacy would be a severe challenge. Large amounts of data are generated by edge devices, which provides an intruder the ability to perform privacy mining, integration, and data association. Therefore, it is essential to create a complex privacy-preserving framework from the perspective of the user's actions, identity, position, and interest.

Although protection of the users' privacy is ensured, a wide variety of data protection functions and safety concerns should be resolved in the context of cooperative contact between users. Privacy-preserving technologies provide benefits for edge applications and service providers, but they impose communicational and computational overhead. The suggested privacy approaches should also satisfy the criteria

of privacy and efficiency and evaluate data analytics security risks in edge computing to offer a few protection and performance criteria.

III. LITERATURE REVIEW

From a holistic viewpoint, Liu *et al.* [12] described the shaping factors of edge computing. They provided a thorough analysis of the concept of edge computing, and the architecture also presents exciting applications for edge computing. There are five essential criteria: availability, confidentiality, privacy requirements, integrity, access control, data protection, and privacy specifications. Next, they described a detailed overview of possible privacy and security issues in edge computing. The current data protection and privacy management frameworks are discussed, and data security research architecture is proposed. The progress of the IoTs around edge computing has been widely researched in recent years. Shi and Dustdar [13] addressed problems such as latency, restricted battery life of mobile devices, security and safety, and provided real-time examples. Security and privacy at the end of the edge network are described. Shi and Cao state that edge computing can resolve battery life issues, response time, bandwidth, cost savings, safety, and data protection. The idea of edge computing is introduced, followed by several studies, ranging from clouds to smart cities and homes, to materialize the concept of edge computing and interactive edges. Ultimately, in edge computing, they present many challenges and opportunities. Shi and Dustdar [13] discussed several shortcomings of edge computing in the IoTs, such as latency, bandwidth, availability, energy, and security, and stated the evolution of edge computing and its basic functionalities. Shi *et al.* [14] defined community problems, from the necessary technology to emerging real-time applications and future business models.

Caching authoritative content at edge nodes at the edge server is a critical approach for improving the efficiency of consumer services in future generation networks. However, it is difficult to correctly forecast potential popularity. Thar *et al.* [15] proposed a caching scheme involving predicting each content's future class label, caching the expected content with high popularity scores, and the future popularity score of content on the basics of projected class labels. Parallel disk access technologies have resolved the speed disparity between disk access and processors by relying on input and output technologies and the cache. Karedla *et al.* [16] looked at caching to minimize device response times and improve disk system data throughput. Kotz and Ellis [17] suggested a parallel access approach that performs load balancing dynamically across every disk depending on each disk load volume. Big write caches are commonly used to increase the throughput of the storage device by leveraging spatial and temporal locations, i.e., data may be combined and overwritten many times until written to the disk. Kotz and Ellis [17] researched numerous cache-control methods used in parallel systems to bridge the disk and processor speed difference. Rajasekaran [18] proposed a randomized and deterministic algorithm for a parallel disk system. Despite enhancements to their disk system, many of the current implementations do not improve the security efficiency of storage systems. Our thesis attempts to explore the use of cache to enhance the security and efficiency of parallel system servers. By introducing hybrid storage systems and new dynamic solid-state disk partitioning, the level of security can be increased.

IV. THE PROPOSED ARCHITECTURE AND DATA MANAGEMENT PLAN

A. THE PROPOSED ARCHITECTURE

In this section, we propose a novel architecture called Secure-Stor and a set of its supporting features is explained, including data partitioning, security management, and data placement. Fig. 1 illuminates the hybrid architecture for Secure-Stor. CDN is responsible for storing metadata and stream videos. The proposed architecture consists of an array of hard disk drivers (HDDs), an array of solid-state disks (SSDs), a parallel system server, and a data handler request transmitter and receiver that are directly connected to the edge devices. The number of hard disk drives and solid-state disks are independent of each other. An array of SSDs is a nonvolatile cache that will save the highly requested data, and is used to boost performance. If the requested content is not available at the SSDs, it is requested from the HDDs; if there is an error or a failure in the HDDs, then the data is requested from the central location server. The security management controller is the heart of Secure-Stor architecture. It comprises a security middleware service, a solid-state disk partitioning system, a real-time response time estimator, and a security controller. The security middleware services are responsible for assigning a security level for each user request based on the request response time. Security middleware services are incredibly adaptable because they allow new security services, such as new encryption algorithms, to be added or to replace old security services. As shown in Table 1, nine different encryption algorithms have been applied by the security mechanism. In conjunction with the cryptographic algorithms' efficiency. The security level 1 is allocated to the best and slowest cryptographic algorithm.

B. SECURITY MANAGEMENT CONTROLLER

The performance of the whole architecture can be significantly enhanced by employing an array of solid-state disks. Solid-state disks are vital because they can judiciously store frequently used data for future access. Notably, the solid-state drives can help shorten the response time for data requests if they are not available in the edge server, making it possible to increase security for the whole system while meeting the user's deadlines. The least recently used algorithm (LRU) is used for moving data from SSDs to HDDs. To maximize the request security without violating the application deadline, the SSDs are vigorously segregated among the hard disk drives. The total size of the solid-state disks are divided into several equal-size partitions. Each hard disk drive is assigned

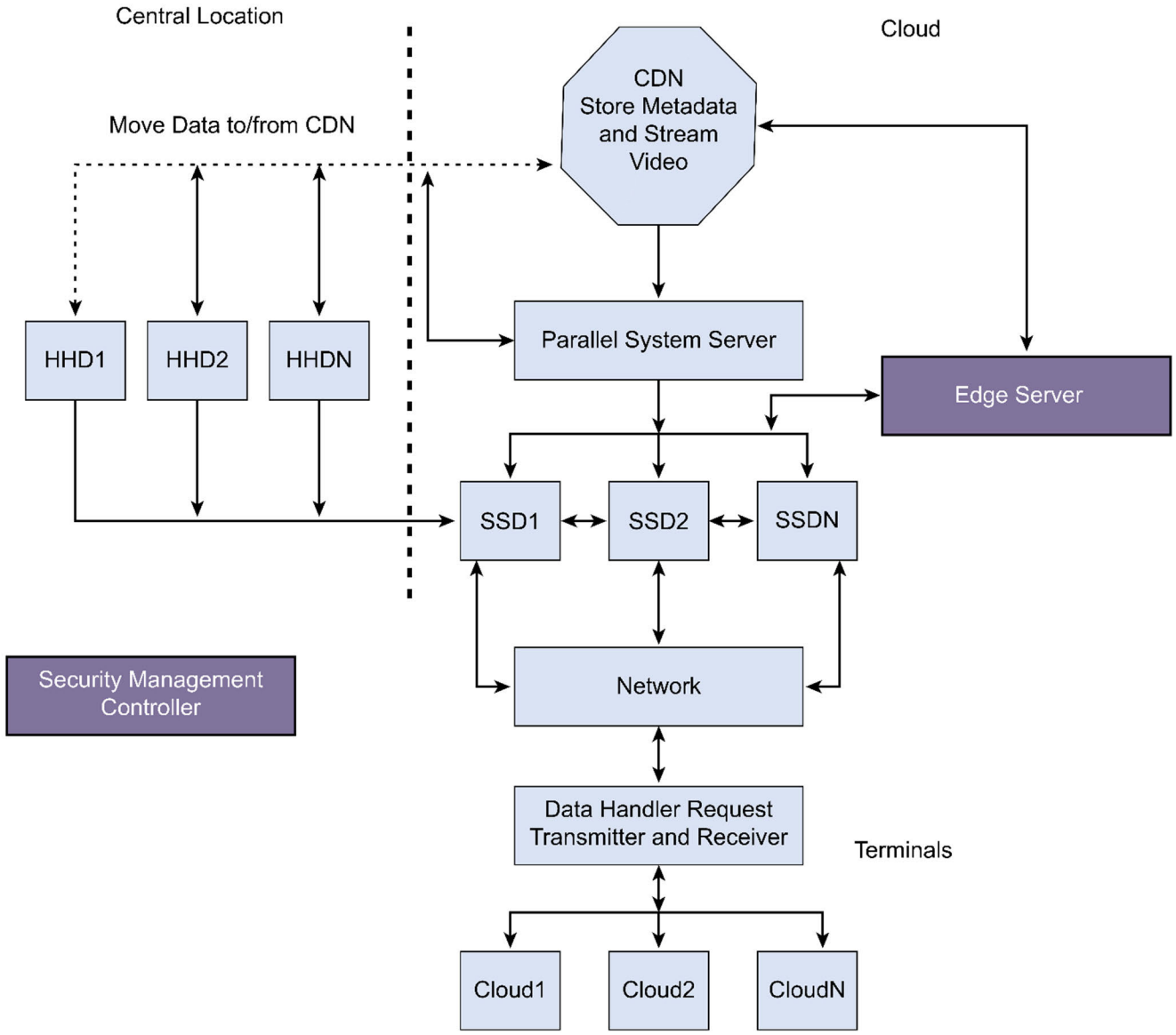


FIGURE 1. The general architecture of secure-stor.

one or more sections based on the hard disk drive workload. Each solid-state disk partition is managed individually using the least recently used replacement policy.

The group of security services that are required by an edge device request Er_i is $S_i = (S_i^1, S_i^2, S_i^3, \ldots\ldots, S_i^j)$, where $\mathbf{S_i^j}$ is the required level of security range for each edge device request.

If we let PA_i be the degree of parallelism of Er_i, then, the security allowance of request Er_i can be calculated by the following equation:

$$S\left(Er_i, PA_d\right) = \sum_{j=1}^{PA_m} SecLevels\left(Er_i, PA_d\right), PA_m \leq K, PA_d \leq PA \quad (1)$$

where PA is the total size of the solid-state disks, PA_d is the d^{th} hard disk drive partition size, and K is the number of hard disk drives available in the cloud system. The parallelism degree PA cannot exceed the total number of hard disk drives in the whole system. Our main goal is maximizing the level of security of all requests, which can be formalized by the following nonlinear equation:

$$Maximize \sum_{d=1}^{k} S\left(Er_d, PA_d\right) \quad (2)$$

C. THE RESPONSE TIME ESTIMATOR OF EDGE DEVICES

Edge device request response time is described as the time gap between the request launched by a client or edge device and the time the subsequent input/output operations are

TABLE 1. Cryptographic algorithms for confidentiality services [19].

S. No.	Cryptographic algorithms	SI^c_i: Security level	μci: KB/ms
1	SEAL	0.1	15.9
2	Blow Fish	0.2	13.2
3	RC5	0.3	9
4	DES	0.4	7.9
5	CAST-128	0.5	5.6
6	IDEA	0.6	3
7	AES	0.7	2.2
8	3-DES	0.9	2
9	RSA	1.00	1.5

performed by the parallel system server. To improve edge request security levels, an adaptive solid-state disk partitioning scheme is developed for each disk need and an estimate of the edge request response time is determined. When a freshly issued edge request Er arrives, the response time of Er can be calculated by:

$$T\left(E_r, PA, SL\left(r\right)\right) = T_{queue} + {}^{p}max_{i=1}\left\{T^i proc\left(E_r, PA_{d(i)}, SL\left(E_r, PA_{d(i)}\right)\right)\right\} \quad (3)$$

where PA is the degree of parallelism; Tqueue is the queueing waiting delay at the edge device; $PA_{d(i)}$ corresponds to the HDD partition volume of the disk serving the i^{th} stripe unit; $SL(Er_i, PA_{d(i)})$ is the security level, as shown in equation (3), by the i_{th} tripe unit of edge request Er, SL(r) = (SL(Er_1, $PA_{d(1)}$); and SL(Er_2, $PA_{d(2)}$ . . ., SL(Er_p, $PA_{d(p)}$)) is the security level f the disk request for its p stripe units. T^i *proc* is a delay experienced in system processing by the i^{th} stripe of the Edge request Er. The processing delay of the system T^i *proc* for the request of the i^{th} stripe can be defined as:

$$T^i_{proc}\left(Er, PA_{d(i)}, SL\left(Er_i, PA_{d(i)}\right)\right) = T^i_{security}\left(Er, PA_{d(i)}, SL\left(Er_i, PA_{d(i)}\right)\right) + T^i_{ntework}\left(Er\right) + T_{SSD\,data} + h^i_{SSD\,data}\left(Er, PA_{d(i)}\right) T^i_{HHD\,disk}\left(Er\right) + h^i_{HHD\,data}\left(Er, PA_{d(i)}\right) T^i_{CDN} \quad (4)$$

where $T^i_{network}$, $T^i_{security}$, $T_{SSD\,data}$, $T_{HHD\,data}$, and T^i_{CDN} are the delays at the network subsystem, security mechanism, SSD data, HHD data access, and content Accesses from CDN, respectively. The $h^i_{SSD\,data}(Er, PA_{d(i)})$ in the previous equation specifies whether the data are retrieved from SSD or HDD.

$$h^i_{SSD\,data}\left(Er, PA_{d(i)}\right) = \begin{cases} 0 & \textit{if there is data found in SSD} \\ 1 & \textit{if the data is dot found in SSD} \end{cases}$$

Similarly, the $hi_{HDD\,data}(Er, PA_{d(i)})$ term in the above equation indicates whether the data are retrieved from HDD or the CDN.

$$h^i_{HHD\,data}\left(Er, PA_{d(i)}\right) = \begin{cases} 0 & \textit{if there is data found in HDD} \\ 1 & \textit{if the data is dot found in HDD} \end{cases}$$

The security overhead, which is the delay in processing at the security mechanism, will be subject to the assigned security level (SL) and divided partitions across the SSDs. As each algorithm's level of security increases, the efficiency decreases. The size of data secured and algorithms (cryptographic) used for encryption are responsible for computational overhead due to encryption. Based on the reference paper [20], $T^i_{security}$ can be easily derived (Er, PA, σ) from the following:

$$T^i_{security}\left(Er, PA_{d(i)}, SL\left(Er_i, PA_{d(i)}\right)\right) = T_{security}\left(SL\left(Er_i, PA_{d(i)}\right)\right) = \sum_{k=1}^{q} \frac{d}{p\mu^k\left(s^k_i\left(PA_{d(i)}\right)\right)} \quad (5)$$

where d refers to the size of data requested, i.e., data size, and d/PA is defined as the i^{th} data size divided across a set of HDDs or SSDs. The security service throughput of whose security level ($s^k_i(PA_{d(i)})$ is defined as $\mu^k(s^k_i(PA_{d(i)}))$. Based on the previous equation, the security overhead model shown in the security management controller explains the total processing delay due to security mechanisms. The overhead because of a specific security service is defined as the ratio of the data size and throughput of the corresponding security service.

$$T^i_{network}\left(Er\right) = \frac{i\frac{d}{PA_i} + \sum_{j=1}^{k} d_j}{B_{network}} \quad (6)$$

The $B_{network}$ represents the effective bandwidth of the network, and PA_j represents the j^{th} hard disk data size in the network queue.

For the i^{th} stripe unit of the edge request, when arriving at the HDD disk, disk j should process x disk requests before handling the arrived edge request. Therefore, the delay in the HDD subsystem Ti_{disk} is given as:

$$T_{HDD\,disk}\left(Er\right) = T_{HDD\,disk,j}\left(\frac{d}{PA}\right) + \sum_{l=1}^{x} T_{HDD\,disk,j}\left(d\right) \quad (7)$$

where $T_{HHD\,disk,j}(d)$ is defined as the time required to process requests of d bytes of data in an HDD subsystem.

$T_{HDD\,disk,j}(d)T_{HDD\,disk,j}(d)$ can be written as:

$$T_{HDDdisk,j}(d) = T_{rotation} + T_{seek} + \frac{d}{B_{disk}} \tag{8}$$

where $T_{rotation}$ and T_{seek} are the rotational latency and seek time, and $\frac{d}{Bdisk}$ is the data transferring time depending on the disk bandwidth B_{disk} and data size d.

Similarly, when the edge request K^{th} stripe unit arrives at the CDN when the requested data are not available in the HDD disk, the data are fetched from the CDN. There are y Edge requests that must be processed by the CDN before handling that Edge device request. Thus, the delay in the CDN subsystem T_{CDN} is given as:

$$T_{CDN}(Er) = T_{CDN}(\frac{d}{PA}) + \sum_{l=1}^{k} T_{CDN}(d)$$
$$T_{CDN}(Er) = T_{CDN}\left(\frac{d}{PA}\right) + \sum_{l=1}^{k} T_{CDN}(d) \tag{9}$$

where $T_{CDN}(d)$ is the CDN processing time of a request containing d bytes of data, and T_{CDN} can also be derived as:

$$T_{CDN} = T_{round\,trip\,time} + T_{latency} \tag{10}$$

where $T_{round\,trip\,time}$ is the delay in time when an edge device requests to be transferred from a starting point to a destination and back again to the starting point, and $T_{latency}$ is the time that passes between user action and the resulting response due to network delays or by an Internet delay.

D. DATA MINING IN EDGE COMPUTING

To make a significant decision to predict content popularity based on end-user expectations and decide which contents should be stored in the SSDs and HDDs, deep learning models are trained as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and long short-term memory recurrent neural networks (LSTM RNNs). These are used in the cloud center or CDN. The data predicted by the learning model are then sent to the parallel system server. From there, data are distributed to the SSD and HDD to store the desired contents. The algorithm's fundamental aim is to reduce the delay in the accessibility of content to the user by storing frequently used contents at SSD and HDD with the use of deep learning content-based popularity algorithms. To forecast future popularity content and the future demand counts, the supervised-depth learning approach is used for training. The contents are then cached, depending on expected results. In an edge network, the process of supervised learning should include the following steps. First, user content ratings and preferences are collected during peak activity hours. Supervised learning algorithms estimate content popularity. The most common content for a group of edge users is cached. In the case of a cache hit, the user request is served from the local cache.

The learning process should forecast patterns unique to the edge network while simultaneously satisfying storage constraints at the edge. With little to no knowledge, ML techniques learn and can identify popular content with time variance enabled by data-driven analytics. Caching and distribution problems can be configured for a collection of network states and storage constraints. An appropriate prediction model is challenging to find among the different types of deep learning models, such as convolutional neural networks (CNNs), multilayer perceptron's (MLPs), and recurrent neural networks (RNNs).

A content delivery network (CDN) is performed as the high-end computing server, and the edge server or parallel system server is implemented as the base station. Using gathered data from the edge server, the CDN is responsible for deep learning model training and then to forecast the content's potential importance with trained models and content sent to the SSD and HDD to store common content. Contents predicted by the CDN are responsible for storing SSDs and HDDs. If the edge device requested Data are found in the SSD and HDD, the network instantly responds to the end-users. If not, the requested data is obtained from the CDN and sent to the users.

As shown in Fig. 2, the CDN contains the Data Module, Management Module, Generating Module, Preprocessing Module, Prediction Module, and Training Module. The management module is responsible for controlling all components of the CDN and parallel system server. The Data Collector Module collects data that are frequently requested. The preprocessing module removes unwanted data, extracts the log files and forms the prediction model dataset. The generative module produces different prediction models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) [21]. The training module controls the training phase of the constructed models using a dataset and stores relevant details such as accuracy, validation accuracy, and model configuration. Then, the management module chooses the most refined model and manages the prediction model's training, starting from where to store the state of the trained model and when to end training by monitoring the training module.

Furthermore, the management module transfers and generates the content containing predicted content and request counts to the parallel system server. The parallel server consists of a data collector module and a set of SSD and HDD modules. The Data Collector collects the number of requests and hit count. The parallel system server downloads content from a server based on the content generated and delivered by the CDN. The Data Handler Request Transmitter and Receiver controls and organizes the arriving requests from edge devices. The arrived request is checked by the request handler to determine whether the content that is requested is present in SSD or HDD. If the requested content is in the SSD or HDD, then the content is delivered to the edge device; otherwise, the Data Handler accesses the content from the CDN.

E. LONG SHORT-TERM RECURRENT NEURAL NETWORK (LSTM RNN)

Recurrent neural networks can scale deep network models for approximation functions to manage temporal sequences.

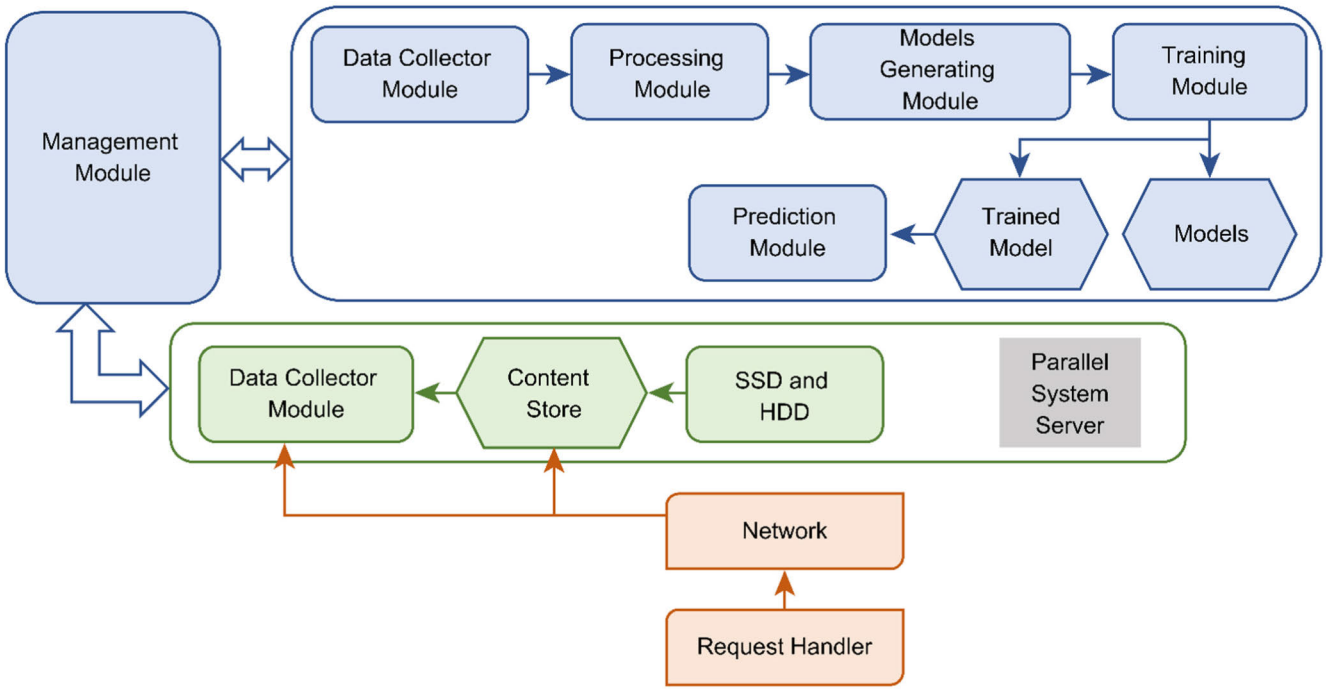


FIGURE 2. An overview of the learning-based model.

The output from the previous time step determines the decision that the network produces for the next step. In addition to the current model input, RNNs involve the use of memory to recall data acquired in previous time measures. There are different types of RNNs, such as gated recurrent units and LSTM. LSTM [22] is chosen because it can provide better output than regular recurrent units using LSTM cells. LSTM networks are well suited for making predictions, classifying, and processing based on time series data. The memory cells are extended along with RNNs to provide the output and store information, which helps to learn temporal connections over long time periods.

Running gradient descent-based RNN training created problems with exploding gradients, which were resolved by LSTM models with additional information flow structures called gates. The LSTM model has proven accurate and useful in solving mobility problems and traffic prediction within communication networks. The main objective of LSTM is problem optimization, popularity prediction, and mobility, and its patterns are from history-based locality and user context-based popularity. It improves the cache hit ratio and reduces transmission energy and delay.

As shown in Fig. 3, the full LSTM-based prediction model contains a set of input layers, which needs feed input, i.e., a batch normalization layer that allows even higher learning speeds. The time series data are learned by recurrent layers to avoid overfitting issue dropout, whereas time distributed network layers can also classify the output along with the data layers. The network is trained by backpropagation. The predicted output values can be iteratively updated to the output layer.

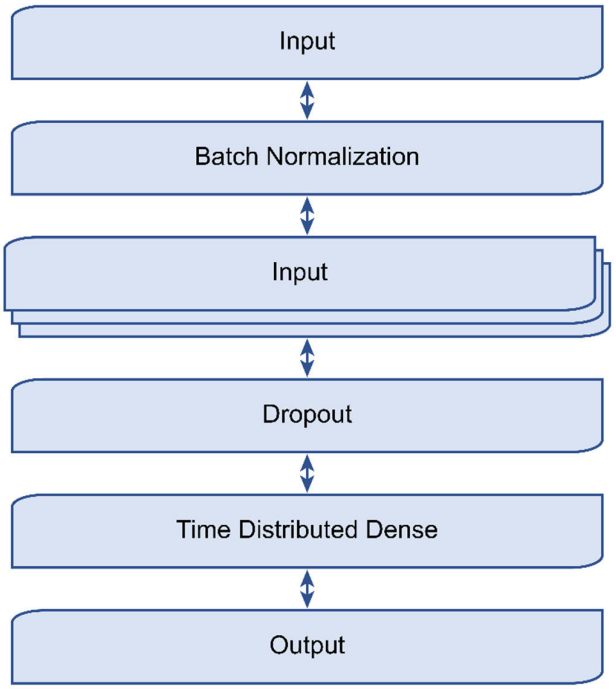


FIGURE 3. Recurrent neural network model.

The training process for predicting the content of the RNN model at the CDN is to improve the accuracy of the model. The inputs for this algorithm are a subset of raw data obtained from edge devices. First, the training accuracy and validation accuracy metrics of the RNN model are assigned a value of zero. Then, the regularization rate and learning

Input:
ER : A newly arrived edge device request
D: Absolute deadline of the request
S_1: The lower bound of the edge device level of security
S_m: The upper bound of the edge device level of security
R: Waiting queue for the edge device request
1. Insert the requests into R based on their absolute deadline
2. Partition the solid state disks among the hard disk drives based on the hard disk drives workload
3. For each request do
4. Initiate the lower bound of the level of security for each request to 0.1 (0.1 means use a weak encryption algorithm)
5. While (D_i < desired response time) do
6. If the level of security < 1 (1 is the strongest encryption algorithm)
7. Increase the level of security by 0.1 (go to the next stronger encryption algorithm)
8. End while
9. If D_i > desired response time
10. Then decrease the level of security by 0.1 (go to the weaker encryption algorithm)
11. Deliver the requests

FIGURE 4. The proposed secure-stor algorithm.

rate are assigned fixed values. The finest values are found for the regularization rate and learning rate by iteration of input data. In this process, the solution's behavior changes are continuously updated with input conditions. If the old validation accuracy is less than the current accuracy, then the regularization rate and learning rate are updated. The output of this RNN algorithm is an optimized trained model for predicting popularity content, which is then transferred to SSDs and HDDs by the parallel system server. The LSTM model architecture outperforms most of the algorithms on all crucial metrics, i.e., training time, training accuracy, and validation.

The training process for predicting the content of the RNN model at the CDN is to improve the accuracy of the model. The inputs for this algorithm are a subset of raw data obtained from edge devices. First, the training accuracy and validation accuracy metrics of the RNN model are assigned a value of zero. Then, the regularization rate and learning rate are assigned fixed values. This is completed to find the finest values for the regularization rate and learning rate by iteration of input data. In this process, the solution's behavior changes are continuously updated with input conditions. If the old validation accuracy is less than the current accuracy, then update the regularization rate and learning rate. The output of this RNN algorithm is an optimized trained model for predicting popularity content, which is then transferred to SSDs and HDDs by the parallel system server. The LSTM model architecture outperforms most of the algorithms on all crucial metrics, i.e., training time, training accuracy, and validation.

V. THE PROPOSED ALGORITHM

The flow representation of the Secure-Stor algorithm is shown in Fig. 5. Consider a newly arrived Edge Device

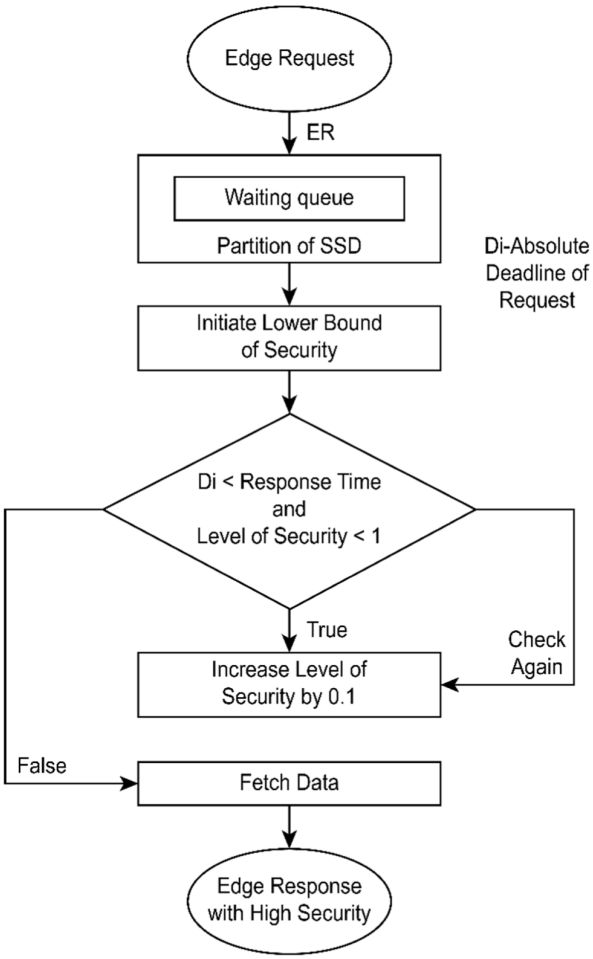


FIGURE 5. Flow chart representation of the secure-stor algorithm.

Request as Er_1. The absolute Deadline of the request Er_1 is D1. The best encryption algorithm is chosen for data transfer with maximum security and completes the given request within the deadline using our Secure-Stor algorithm. Each request has a different security level based on the absolute deadline and encryption algorithm we choose. First, a weak encryption algorithm measured as 0.1 Complexity is used and determines whether the given request response can be finished within the absolute deadline. If that satisfies the given deadline, it can further increase the complexity of the encryption algorithm by 0.1 and determine whether the request-response can be finished in the absolute deadline; if this is satisfied, then the process can be continued by increasing the complexity of the encryption algorithm every time by 0.1, where the most robust encryption algorithm complexity is one that compares with the deadline until the request is assigned with maximum security and the response is sent in the deadline without delay. The New Edge Request is —Er1, and the Absolute Deadline is –0.7 Nano Seconds.

Here, the encryption algorithm for request was chosen based on the response time and deadline. Assuming the Weak Encryption Algorithm S1 with a 0.1 complexity and a response time of 0.2, then (0.7 < 0.2); therefore,

if we increase the complexity by 0.1 with Encryption Algorithm S2, the new response time is 0.4, and still the response time is below the Deadline ($0.7 < 0.4$), then the complexity of an algorithm is further increased using S3 and the response time becomes 0.6, which satisfies the Deadline ($0.7 < 0.6$). If the complexity is increased using S4 Encryption Algorithm the response time will become 0.9, which does not meet the Deadline of 0.7. Therefore, this would be terminated and the S3 encryption algorithm would be used, which has maximum security and completes within the deadline.

Theorem: O(nm) is the time complexity of the Secure-Stor algorithm, where m is the maximum security level, and n is the number of edge requests in the queue.

Proof: The security level of each edge request can be increased in $O(m)$ time. Since there are n edge requests with $O(n)$ time at the waiting queue, the time complexity is $O(n)O(m) = O(nm)$ to improve the security of all disk requests.

A. THE OPTIMALITY OF THE PROPOSED ALGORITHM

In this segment, the concept of nonsecure response time is used. Illustrate the optimality of Secure-Stor in terms of the security efficiency of each edge device request. The time of the stripe unit to respond that is not secured for the j^{th} stripe unit in edge device request Er_i is described as the nonsecure time $T^{ij}_{nonsecure}$.

Therefore, the following equation applies:

$$T^{ij}_{nonsecure} = T_{queue} + T_{partition} + T_{SSD\,data} + T_{HHD\,data} + T^{ij}_{disk} \tag{11}$$

Theorem 1: For edge device request Er_i corresponding to any given stripe unit (j^{th} stripe unit) and a security management controller, the Secure-Stor algorithm tends to maximize SL σ_{ij} of the corresponding stripe unit.

Proof: Let us consider the j^{th} stripe unit in the Edge device request Er_i without loss of generality. The theorem is proved by showing to the impossibility to further raise the security level σ_{ij} of the j_{it} stripe unit. The first few steps in the algorithm try to reduce the nonsecure response time of all stripe units in the edge device request Er_i using solid-state disk partitioning.

Accordingly, the nonsecure time of response of any stripe unit is reduced. Thus, the nonsecure time to respond $T_{queue} + T_{partition} + T_{SSD\,data} + T_{HHD\,data} + T^{ij}_{network} + T^{ij}_{disk.}$ of the j^{th} stripe, the unit is reduced. Since Secure Stor cannot decrease the T queue + T partition, the algorithm decreases $T^{ij}_{network} + T^{ij}_{HHD\,data} + T^{ij}_{SSD\,data}$.

Step 5 ensures that the equation $T_{queue} + T_{partition\,d} + T^{ij}_{network} + T^{ij}_{SSD\,data} + T^{ij}_{HHD\,data} + T^{ij}_{security} \leq t_i$ holds. Accordingly, $T^{ij}_{security} \leq t_i - (T_{queue} + T_{partition} + T^{ij}_{network} + T^{ij}_{SSD\,data} + T^{ij}_{HHD\,data})$. The equation $T_{queue} + T_{partition} + T^{ij}_{network} + T^{ij}_{SSD\,data} + T^{ij}_{HHD\,data}$ is the first phase of the algorithm that minimizes the inequality's right-hand side, meaning that $T^{ij}_{security}$ spent in security is increased. As an outcome, steps 6 and 7 of the algorithm exploit the increased time $T^{ij}_{security}$ to improve security σij. Thus, Theorem 1 is complete.

Theorem 2: For an edge device request Eri, if the levels of security of all corresponding stripe units are maximized, then the secure-stor algorithm improves the security quality for an edge device request Er_i when written.

$$\forall \leq j \leq P_i : \sigma_{ij}\ is\ maximized \rightarrow \sum_{i=1}^{p_i} \sigma_{ij}\ is\ maximized \tag{12}$$

Proof: Without effects on the security level of the other strip of Eri, the value of σ_{ij} can be maximized. The SLs of all the stripe units inside the edge request Eri can be increased by the Secure-Stor algorithm simultaneously. Consequently, the summation of security levels of all the stripe units of the edge device request Er_i has increased by steps 6 and 7., i.e., $\sum_{j=1}^{pi} \sigma_{ij}$ is maximized. Thus, the theorem is proved.

The theorem below shows the Secure-Stor algorithm's optimality.

Theorem 3: For any given Edge request Er_i and Security Management controller, the optimization of the quality of Security for Edge device request Eri is done by Secure-Stor

Proof: Theorem 1 states that every stripe unit in the Edge device request Er_i has the respective level of security increased. Theorem 2 proves that if the levels of security of all the stripe units in an Edge device request Er_i are increased in a security management controller, then the Secure–Stor algorithm improves security quality for edge device request Er_i. Therefore, from Theorems 1 and 2, the proof is immediate.

VI. EXPERIMENTAL RESULTS

To test the Secure-Stor framework's performance, a simulation toolkit consisting of a collection of critical components was implemented. The core component includes parallel data transfer between the HDDs and SSDs and parallel data transfer between the edge server, CDN, and the solid-state disks. Nine confidentiality services algorithms are implemented within the simulation toolkit. Eight SSD disks, each with a capacity of 1 GB, and eight hard disk drives with 200 GB are used. Comparably, other frameworks do not incorporate the hybrid storage system in the central location and do not require solid-state disk partitioning. In our toolkit, two metrics are used to show the efficiency of Secure-Stor. The first metric is the success ratio, which is the fraction of total arrived edge server requests that are completed before their absolute deadline. The second metric is the average security level, which is the ratio between the sum of the security level of all requests and the total requests.

A. IMPACT OF ARRIVAL RATE

In this study, the impact of the edge device requests arrival rates when the solid-state disk size and the disk bandwidth are varied. To accomplish this goal, the arrival rate was increased from 0.1 to 0.5 NO./SEC. The edge device request data size is set to 1 GB, and the disk bandwidth is set to 100 MB. The Secure-Stor algorithm reveals that it significantly

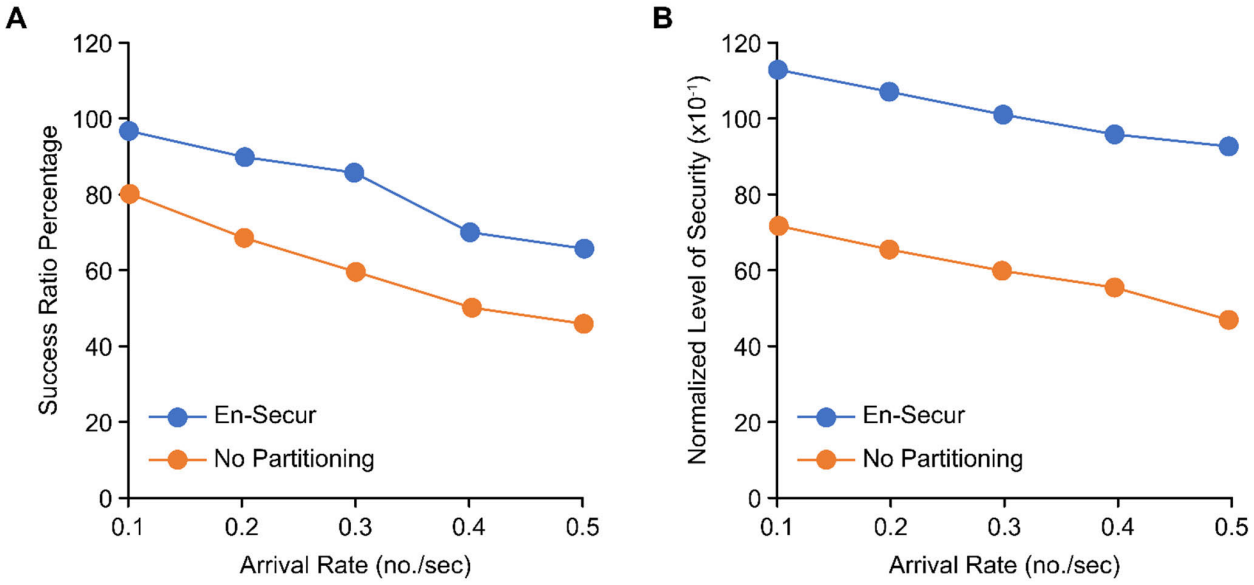


FIGURE 6. (a) Impact of arrival rate on satisfied ratio. (b) Impact of arrival rate on security level.

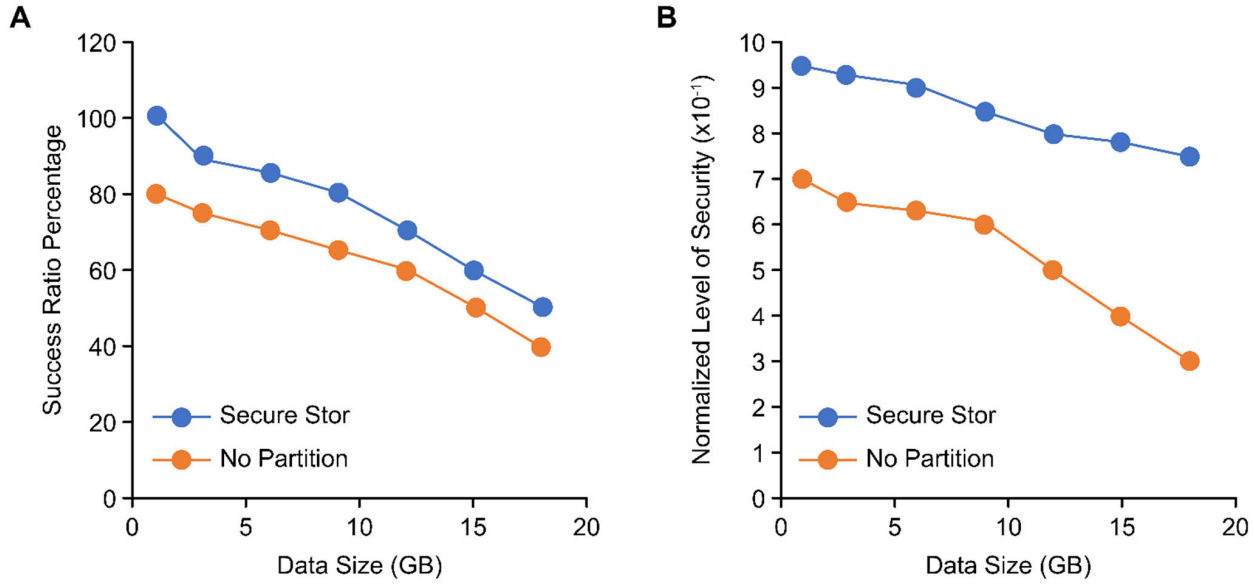


FIGURE 7. (a) Impact of data size on satisfied ratio. (b) Impact of data size on security level.

outperforms the other algorithms. Additionally, Fig. 6(a) and (b) show that Secure-Stor outperforms different algorithms that do not use the component proposed in the Secure-Stor framework. Secure-Stor delivers a higher level of security than the other algorithms.

B. IMPACT OF THE EDGE DEVICE REQUEST DATA SIZE

This section analyzes the efficiency impacts of the requested data size on a parallel system server by differing the data size from 1 GB to 20 GB while keeping all other workload parameters the same. Fig. 7(a) and (b) reveal the impact of data sizes on a parallel server with respect to success ratio percentage and normalized level of security by comparing to the server that does not use hybrid storage architecture and solid-state disk partition. As the data size increases from the graph, a higher success ratio is observed compared to the edge server that does not implement hybrid storage with parallel data transfer with maximum security.

C. IMPACT OF THE EDGE DEVICE REQUEST WITH RESPECT TO DISK BANDWIDTH

In this experiment, we test the impact of the disk bandwidth in a parallel system server. The value of the bandwidth varies from 10 MB/sec. to 100 MB/Sec. Fig. 8(a) reveals that the satisfying success ratio of the Secure-Stor architecture, which uses hybrid storage and dynamic solid-state

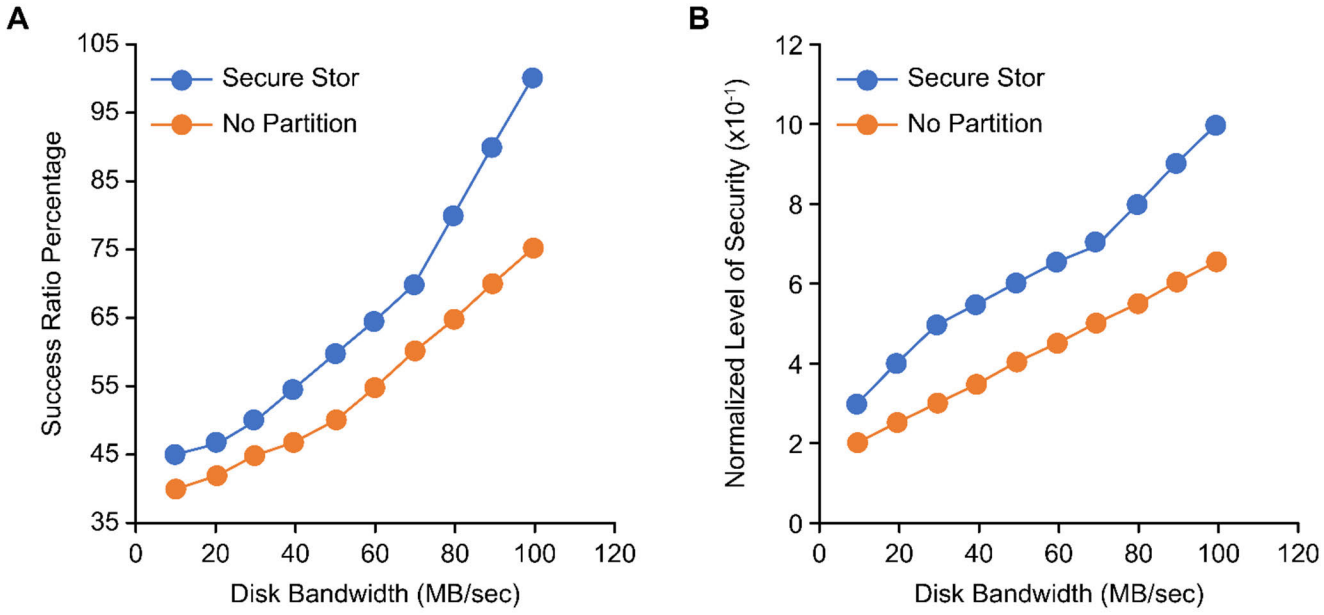


FIGURE 8. (a) Impact of disk bandwidth on satisfied ratio. (b) Impact of disk bandwidth on security level.

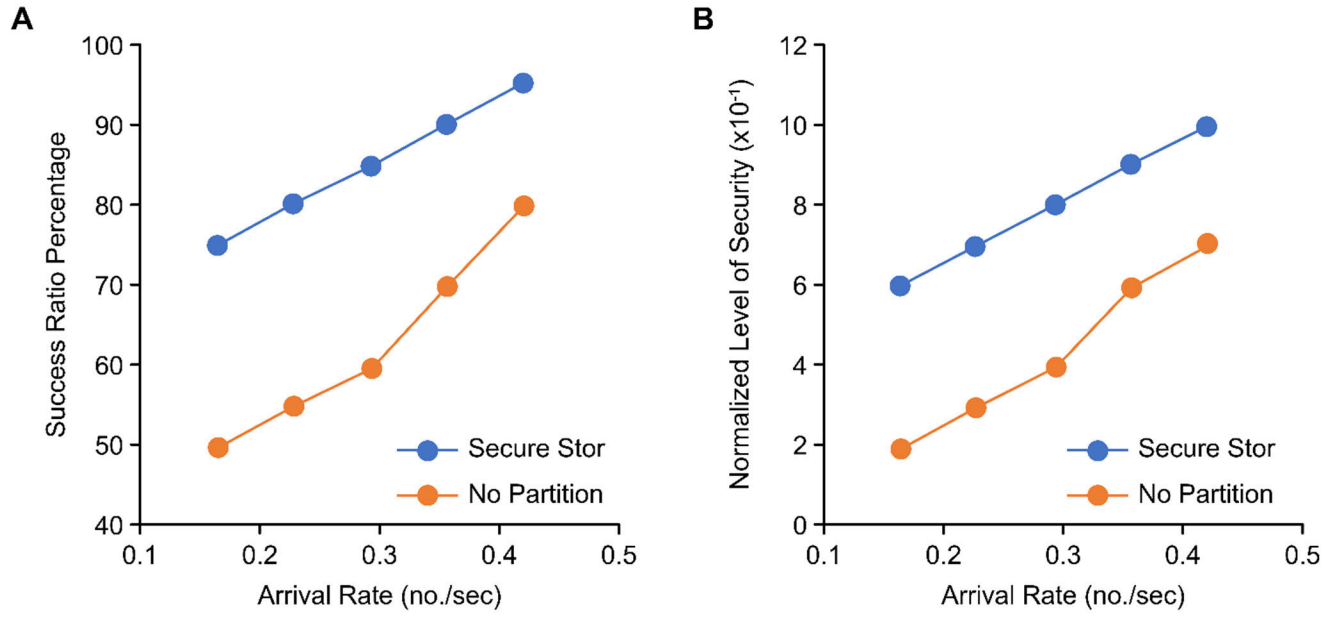


FIGURE 9. (a) Impact of the number of SSDs on the satisfied ratio. (b) Impact of the number of SSDs on the security level.

disk partitioning, is improved as the bandwidth of the disk increases. High disk bandwidths allow short data transmission times, which lead to lower processing times for edge device requests. Consequently, several disk requests can be finished within their absolute deadline. As the bandwidth of the disk increases, the disk processing time decreases, and the security level increases for a particular edge request. Therefore, the Secure-Stor algorithm outperforms the other algorithm, which does not use the component proposed in the Secure-Stor framework at the security level as observed in Fig. 8(b).

D. THE IMPACT OF THE NUMBER OF SOLID-STATE DISKS

The impact of performance by varying the number of solid-state disks from 4 SSD to 20 SSD with 4 SSD. Fig. 9(a) shows that as the number of SSDs increases, the satisfied success ratios of the Secure-Stor algorithm also increase due to more frequent data stored in the SSD, leading to lower data access from the CDN. Additionally, Secure-Stor outperforms the other algorithm, which does not use hybrid storage to satisfy the success ratio. Fig. 9(b) shows that the increasing number of SSDs helps increase the security levels. The overall performance of Secure-Stor is the best

among the algorithms, especially with a higher number of SSDs.

VII. CONCLUSION

Approximately 75% of companies have considered processing data at the edge by 2025, and the edge of the network is a new hotspot for IT investments for an increasing number of organizations. Edge computing allows organizations to enhance processing, analytics capabilities, and data storage. With the transfer of data from network sources to edge platforms, security risks emerge. The edge serves as a primary target for hackers because the edge architecture connects with hundreds of thousands of network-connected computers.

This paper presents a novel hybrid framework that integrates CDN with edge servers in the cloud and a hybrid storage architecture that consists of hard disk drivers and solid-state discs in centralized storage. Consequently, security has become crucial for cloud and edge computing. With our proposed Secure-Stor algorithm, hybrid storage systems, and new dynamic solid-state disk partitioning, the level of security is increased, and the security middleware services are extremely configurable that allows new security services, such as new encryption algorithms to be added or to replace old security services with new ones. As with ever-growing technology, different encryption algorithms are introduced with higher security and better performance by replacing them, making the edge request even more secure in the future. Our simulation results reveal that when comparing the performance and the security of Secure-Stor with another architecture that does not use the solid-state partitioning technique, Secure-Stor significantly increases the security and performance of the system by an average of 85%.

Our future work will focus on determining secured and suitable models among various types of deep learning models and characterizing cryptographic algorithms by high communication and computation costs due to their large key size. Hence, the security management controller should focus on lightweight security, such as block-ciphers lightweight cryptography and permutation-based lightweight cryptography.

REFERENCES

[1] L. Ling, M. Xiaozhen, and H. Yulan, "CDN cloud: A novel scheme for combining CDN and cloud computing," in *Proc. 2nd Int. Conf. Meas., Inf. Control*, Harbin, China, Aug. 2013, pp. 687–690.

[2] M. Satyanarayanan and W. Shi, "Overview of edge computing," IEEE Course, Tech. Rep., 2018.

[3] T. Voigt and C. Rohner, "What is the Internet of Things: An introduction," IEEE Course, Tech. Rep., 2017.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016, doi: 10.1109/JIOT.2016.2579198.

[5] A. Koike and Y. Sueda, "Contents delivery for autonomous driving cars in conjunction with car navigation system," in *Proc. 20th Asia–Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Matsue, Japan, 2019, pp. 1–4.

[6] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, "Data security and privacy-preserving in edge computing paradigm: Survey and open issues," *IEEE Access*, vol. 6, pp. 18209–18237, Mar. 2018, doi: 10.1109/ACCESS.2018.2820162.

[7] Y. Xing and H. Seferoglu, "Predictive edge computing with hard deadlines," in *Proc. IEEE Int. Symp. Local Metrop. Area Netw. (LANMAN)*, Washington, DC, USA, Jun. 2018, pp. 13–18.

[8] J. Shujaa, K. Bilal, E. Alanazi, W. Alasmary, A. Alashaikh, and A. Y. Zomaya, "Applying machine learning techniques for caching in edge networks: A comprehensive survey," 2020, *arXiv:2006.16864*. [Online]. Available: https://arxiv.org/abs/2006.16864

[9] M. McClellan, C. Cervelló-Pastor, and S. Sallent, "Deep learning at the mobile edge: Opportunities for 5G networks," *Appl. Sci.*, vol. 10, no. 14, p. 4735, Jul. 2020, doi: 10.3390/app10144735.

[10] Y. Wang and V. Friderikos, "A survey of deep learning for data caching in edge network," *Informatics*, vol. 7, no. 4, p. 43, Oct. 2020, doi: 10.3390/informatics7040043.

[11] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2nd Quart., 2020, doi: 10.1109/COMST.2020.2970550.

[12] D. Liu, Z. Yan, W. Ding, and M. Atiquzzaman, "A survey on secure data analytics in edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4946–4967, Jun. 2019, doi: 10.1109/JIOT.2019.2897619.

[13] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, May 2016, doi: 10.1109/MC.2016.145.

[14] W. Shi, G. Pallis, and Z. Xu, "Edge computing [scanning the issue]," *Proc. IEEE*, vol. 107, no. 8, pp. 1474–1481, Aug. 2019, doi: 10.1109/JPROC.2019.2928287.

[15] K. Thar, N. H. Tran, T. Z. Oo, and C. S. Hong, "DeepMEC: Mobile edge caching using deep learning," *IEEE Access*, vol. 6, pp. 78260–78275, Dec. 2018, doi: 10.1109/ACCESS.2018.2884913.

[16] R. Karedla, J. S. Love, and B. G. Wherry, "Caching strategies to improve disk system performance," *Computer*, vol. 27, no. 3, pp. 38–46, Mar. 1994, doi: 10.1109/2.268884.

[17] D. Kotz and C. S. Ellis, "Caching and writeback policies in parallel file systems," *J. Parallel Distrib. Comput.*, vol. 17, nos. 1–2, pp. 140–145, Jan. 1993, doi: 10.1006/jpdc.1993.1012.

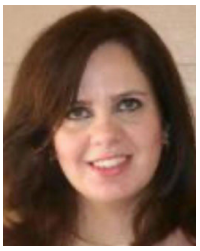
[18] S. Rajasekaran, "Selection algorithms for parallel disk systems," in *Proc. 5th Int. Conf. High Perform. Comput.*, Madras, India, 1998, pp. 343–350.

[19] P. Semwal and M. K. Sharma, "Comparative study of different cryptographic algorithms for data security in cloud computing," in *Proc. 3rd Int. Conf. Adv. Comput., Commun. Automat. (ICACCA) (Fall)*, Dehradun, India, Sep. 2017, pp. 1–7.

[20] M. Nijim, X. Qin, T. Xie, and M. Alghamdi, "AWARDS: An adaptive write strategy for secure local disk systems," in *Proc. IEEE Int. Perform. Comput. Commun. Conf.*, Phoenix, AZ, USA, Apr. 2006, p. 8.

[21] S. Tuli, N. Basumatary, and R. Buyya, "EdgeLens: Deep learning based object detection in integrated IoT, fog and cloud computing environments," in *Proc. 4th Int. Conf. Inf. Syst. Comput. Netw. (ISCON)*, Mathura, India, Nov. 2019, pp. 496–502.

[22] Y. Tian and L. Pan, "Predicting short-term traffic flow by long short-term memory recurrent neural network," in *Proc. IEEE Int. Conf. Smart City/SocialCom/SustainCom (SmartCity)*, Chengdu, China, Dec. 2015, pp. 153–158.



MAIS NIJIM (Senior Member, IEEE) received the B.S. degree in computer science from the Princess Sumaya University for Technology, Amman, Jordan, the M.S. degree in computer science from New Mexico State University, in 2004, and the Ph.D. degree in computer science from the New Mexico Institute of Mining and Technology, in 2007. She has been an Associate Professor with the Department of Electrical Engineering and Computer Science, Texas A&M University-Kingsville, since 2010. Her research interests include machine learning, cyber physical systems, cyber security, and wireless sensor networks. She is an Editor of the *International Journal of Sensor Networks*.

HISHAM ALBATAINEH received the B.S. degree in physics and mathematics from Yarmouk University, Irbid, Jordan, and the master's degree in experimental and high-energy physics and the Ph.D. degree in experimental nuclear physics from New Mexico State University, Las Cruces, NM, USA, in 2005 and 2009, respectively. He has been an Assistant Professor with the Department of Physics and Geosciences, Texas A&M University-Kingsville, since 2015. In addition to nuclear physics, his research interests include smart grids and machine learning.