Secure and Efficient Container Scheduling in Multi-Cloud Using an Elliptic Curve-Based Digital Signature Algorithm

Komala Rangappa
Department of Master of Computer Applications, VTU Research Centre, BMS Institute of Technology & Management, Yelahanka, Bengaluru, India | Department of Computer Applications, M.S. Ramaiah Institute of Technology, Bengaluru, India
komal.uday@gmail.com (corresponding author)

Arun Kumar Banavara Ramaswamy
Department of Information Science and Engineering, BMS Institute of Technology & Management, Yelahanka, Bengaluru, India
arunkumarbr@bmsit.in

Shreyas Arun Kumar
Department of Computer Science and Engineering, Sai Vidya Institute of Technology, Bengaluru, India
shreyasvasista05@gmail.com

Received: 26 August 2025 | Revised: 16 October 2025 | Accepted: 4 November 2025

 DOI: https://doi.org/10.48084/etasr.14329

ABSTRACT

In modern application deployment, containers have become essential due to their lightweight structure and efficient virtualization capabilities. In this domain, container scheduling plays a critical role in effectively assigning workloads to various computing nodes. To address node imbalances and ensure efficient deployment, this study proposes a novel two-phase container scheduling framework that enhances workload allocation and overall system performance. The proposed Makespan-Aware Multi-Task Scheduling with Deadline Restrictions (MAMTS-DR) model treats the scheduling task as a constrained optimization problem, incorporating diverse objective functions aimed at improving server utilization and reducing overall energy usage. To protect the privacy of containers during migration, the model integrates the Elliptic Curve Digital Signature Algorithm (ECDSA) to ensure a secure scheduling environment. In addition, the associated encryption and migration overheads are included in the optimization model. By incorporating container-specific attributes through the proposed attribute-based encryption framework, the proposed approach ensures both security and optimal performance, as the strategic selection of containers and destination nodes further promotes equilibrium in cloud-hosted clusters.

Keywords-containers scheduling; deadline restrictions; digital signature algorithm; elliptic curve; makespan-aware multi-task scheduling

I. INTRODUCTION

Cloud computing has revolutionized distributed systems, turning them into scalable, on-demand platforms that provide computing resources and services over the Internet, tailored to meet the dynamic needs of users [1, 2]. The cloud model is related to distributed computing and involves various virtual computers that are interconnected and dynamic to design computing resources [3]. Cloud computing involves a wide variety of resources, such as networks, physical servers, storage, and applications, allowing users to access them through networks for their service requirements [4]. The National Institute of Standards and Technology (NIST) described cloud computing as a model for allowing universal, suitable, on-demand network access to a shared pool of configurable computing resources, which are often provisioned and released with minimal management effort or service provider interaction [5]. Thus, this framework is developed as a utilization model in which each service, such as computation, storage, and network, is provided to consumers as a service on demand [6]. Cloud computing has increasingly replaced traditional computing models by offering flexible, location-agnostic access to computing power. Leading service providers, such as Amazon Web Services (AWS), Google Compute Engine, Google Cloud, Microsoft Azure, Oracle Cloud, and Rackspace, offer users customizable resources,

including servers, networking infrastructure, storage solutions, and software applications [7]. Cloud services are commonly delivered through three primary models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). IaaS delivers a sophisticated environment, SaaS is a model for providing software and services, and PaaS provides frameworks for development [8-10].

Despite these advantages, security continues to be one of the primary challenges facing the advancement and adaptation of cloud-computing technologies. Security in cloud computing should develop rapidly in terms of remaining up-to-date with newly identified attack vectors [11]. Recent shifts toward multi-cloud environments bring additional layers of complexity, especially in security. In such environments, organizations depend on cloud service providers to avoid vendor lock-in, improve reliability, and reduce costs [12]. Multi-cloud computing presents unique challenges, particularly in optimizing diverse resources distributed across multiple geographic regions [13]. Multi-cloud-based resource allocation is related to different restrictions and diverse configurations, which differ with respect to customer requirements and cloud service providers' schemes [14]. In multi-cloud environments, applications consist of one or more tasks, which are provided as general units of scheduling and implementation [15]. Due to the requirement to employ operations while considering Quality of Service (QoS) necessities, such as resource utilization, energy efficiency, etc., cloud computing faces an important scheduling complexity [16]. In cloud computing, architecture workload balancing is an essential factor that helps to allocate computing resources [17]. The primary objective of service placement is to optimize system performance [18]. The procedure of scheduling and allocating resources, and the advantage of a combined form of resource allocation and task scheduling, is to minimize system latency [19]. Job scheduling is frequently classified as a Non-deterministic Polynomial (NP) hard problem. Identifying an enhanced task scheduling solution within a multi-cloud environment is challenging due to the need to integrate service compositions and align various ideal goals [20].

The key points of this study are:

- Proposes the Makespan-Aware Multi-Task Scheduling with Deadline Restrictions (MAMTS-DR) and Elliptic Curve Digital Signature Algorithm (ECDSA) approach for effective container scheduling in multi-cloud data centers with effective confidentiality, which dynamically makes decisions based on future and existing tasks.
- Integrating container data security into optimization objective functions thoroughly focuses on encryption analysis. The computational overhead of encryption is thoroughly evaluated relative to the capabilities and capacity of the data center machines.
- The effectiveness of the proposed approach is validated using a simulation in the CloudSim toolkit. Performance metrics such as makespan, resource utilization, energy consumption, and Degree of Imbalance (DoI) were used to compare the proposed method with previous works.

II. LITERATURE SURVEY

In [21], an innovative two-phase container scheduling mechanism was proposed to resolve node-level imbalances and promote efficient container deployment. This scheduling model was formulated as an optimization task that incorporates multiple objectives and constraints to improve server consolidation and reduce energy use. Additionally, the model accounts for container confidentiality during migration using encryption techniques, with the associated overhead integrated into the optimization framework. Using attribute-based encryption by considering container characteristics enables secure task deployment across a cloud cluster. In [22], a distributed cloud management solution leveraged blockchain to support seamless sharing of Virtual Machine (VM) metadata across data centers. The BigchainDB platform was utilized to maintain a decentralized ledger that facilitates secure VM scheduling and migration, thus outperforming traditional VPN-based centralized management architectures.

In [23], a framework for elastic Business Process Management (BPM) in multitenant clouds was developed. This solution employs autonomic principles to schedule containers onto available servers and introduces multipath routing to manage intercontainer communication. A multi-objective Crowd Search Optimization (CSO) algorithm assigns containers to appropriate servers, while communication flows are routed using a Discrete Wolf Search Optimization (DWSO) algorithm to construct bandwidth-efficient paths aimed at reducing power usage. In [24], a scheduling framework was based on Decision Functions through Normal Distributions (DFND) to more accurately assess task-to-cloud compatibility. Using multivariate Gaussian models, this model captures resource dependencies such as CPU, memory, and latency. In addition, the Tasmanian Devil Optimization (TDO) approach provides dynamic exploration and exploitation techniques to improve assignment accuracy and scalability in multi-cloud setups.

In [25], an autonomic CSO-ILB load balancer was proposed for multi-cloud environments. This method incorporates a multiloop mechanism to enable self-management before load distribution. An enhanced scheduling algorithm called Deadline-Constrained Makespan Minimization for Multitask Scheduling (DCMM-MTS) was proposed to allocate tasks. According to task assignment outcomes, the CSO-ILB algorithm efficiently balances the container workloads. In [26], a hybrid strategy decomposed the scheduling problem into two levels: (i) clustering servers within each datacenter based on resource usage similarity, and (ii) execution of task allocation and load balancing at the datacenter and cluster levels. This two-stage method supports live deployment and provides high scalability, with interoperability across mechanisms.

In [27], the Fast Privacy-Preserving Single Sign-on (FPRESSO) scheme was proposed to provide an authentication system with anonymous authorization binding for multi-application environments hosted in the cloud. This research utilizes random perturbation to safeguard the SSO token, which is structured as a JSON Web Token (JWT). Essentially, this approach stores tokens in cookies to efficiently manage access

and facilitate SSO recovery. Additionally, this study introduced an anonymous authorization binding protocol to bundle user roles and permissions into the SSO token, enhancing the efficiency and agility of access control across applications. To provide high scalability to accommodate a large number of users in a cloud environment, this study introduced a multithreaded load-balancing algorithm to dynamically handle both SSO token generation and verification requests, ensuring an efficient distribution of load across multiple servers. In [28], a fault-tolerant load-balancing approach used a Multi-objective Cat Swarm Optimization (MCSOFLB) algorithm. The performance of this approach was benchmarked against existing optimization methods, demonstrating improved fault tolerance and workload distribution.

A. Research Gap

Although existing works have advanced container scheduling by focusing on either deadline satisfaction, load balancing, or energy efficiency, they often neglect the overhead introduced by security mechanisms such as encryption and authentication. These methods achieve performance gains, but at the cost of leaving security integration unaddressed. In contrast, security-centric frameworks enhance confidentiality or integrity but lack explicit optimization of makespan or resource utilization. This disconnect highlights the research gap: the lack of approaches that jointly optimize both performance and security. The proposed MAMTS-DR explicitly addresses this gap by embedding ECDSA overhead into scheduling objectives, thereby balancing secure execution with performance efficiency. The proposed MAMTS-DR with ECDSA addresses these gaps by introducing an integrated framework that (i) ensures task execution within deadlines, (ii) reduces overall makespan and energy usage, and (iii) embeds trapdoor-based digital signature authentication to secure container data in transit. This holistic optimization fills a critical void in existing cloud scheduling research.

III. PROPOSED METHOD

The novelty of the proposed Makespan-Aware Multi-Task Scheduling with Deadline Restrictions (MAMTS-DR) method lies in its explicit integration of three aspects that differentiate it from DCMM-MTS and hybrid scheduling methods. First, MAMTS-DR uses a dual-stage optimization strategy that jointly considers task prioritization and resource selection, rather than only focusing on makespan minimization. Second, the model embeds ECDSA-based cryptographic protection directly within the optimization process, thereby addressing security and scheduling as a unified problem while accounting for encryption and migration overheads. Third, the extended Directed Acyclic Graph (DAG)-based scheduling framework enables adaptive multi-cloud deployment, which provides the optimal result in overall performance metrics.

This study introduces a container-scheduling framework combined with multipath routing, tailored for IaaS multitenant cloud environments, that targets two key challenges: container scheduling inefficiencies and excessive power consumption in datacenters. This approach aims to distribute containers across available servers in a way that dynamically adapts to varying workloads, resource demands, and deadline constraints.

- Container Scheduling Strategy: Scheduling is guided by parameters, such as the required CPU, memory, bandwidth, and network resources. The goal is to execute workloads efficiently while meeting user deadlines and maximizing provider benefits. The objectives include minimizing makespan, execution cost, processing time, and improving resource utilization and service income.
- Scheduler Component: A dynamic scheduler is used to manage incoming user tasks, mapping them to suitable containers based on real-time resource needs, ensuring the effective use of the server infrastructure.
- Container Definition: Containers encapsulate the necessary runtime environment and resources (CPU, memory, disk, and network) for task execution. Tasks are deployed in these containers to take advantage of isolated execution units.
- Self-Adaptation Module: This module utilizes the Monitor, Analyze, Plan, Execute – Knowledge (MAPE-K) loop for autonomous management. It monitors the environment, gathers operational insights, and makes adaptive decisions that are stored in a knowledge base for continual learning and policy improvement.
- Evaluation Mechanism: The evaluation unit calculates container utilization and total capacity. Based on these metrics, containers are categorized as balanced, overloaded, or underloaded, which guides further scheduling or migration actions.

In the dynamic workload conditions of a multi-cloud environment, efficiently allocating resources to incoming tasks is essential. The proposed framework employs an extended DCMM-MTS algorithm for deadline-constrained scheduling. This algorithm uses a DAG to structure task-execution dependencies and ensure efficient resource allocation. After scheduling, the system secures container migration using ECDSA, preserving confidentiality and integrity. Figure 1 shows the architecture of the proposed approach.

A. Container Scheduling

The optimal scheduling of containers in cloud environments is formulated as a multi-objective optimization problem, wherein the goal is to execute incoming workloads with high efficiency, minimize delays, and ensure an uninterrupted service. This involves the effective provisioning of available cloud resources. To address this challenge, the proposed MAMTS-DR method is used to identify the most suitable servers for container deployment.

In a dynamic cloud ecosystem, user-submitted tasks arrive continuously and require prompt allocation to available containers within a multi-cloud infrastructure. These tasks typically have specific requirements, such as CPU, memory (RAM), and bandwidth, which are also associated with defined deadlines. The selection of containers for task execution is governed by the availability of resources and task priorities. Containers that were underutilized and capable of meeting the requirements of the task were selected. To enhance scheduling precision, a priority-based task assignment strategy was

applied. The system evaluates multiple clouds to determine where the optimal containers reside and selects the cloud that can offer the minimum makespan for task execution. This uniform scheduling policy was enforced on all participating cloud platforms. In the proposed MAMTS-DR framework, encryption and decryption costs introduced by ECDSA are explicitly embedded in the optimization function rather than being measured only after task execution. These cryptographic overheads are modeled as part of the makespan and energy consumption objectives, ensuring that scheduling decisions inherently reflect the trade-off between security and performance. This integration ensures that the framework simultaneously achieves confidentiality and efficient task allocation, providing a more realistic representation of multi-cloud scheduling behavior.

MAMTS-DR extends conventional deadline- and budget-constrained scheduling models by introducing a makespan-aware optimization strategy that operates in two key stages, task selection and resource selection, ensuring that task execution is aligned with user-defined deadlines while minimizing the makespan and maximizing system efficiency.

- Task Selection: In the initial phase, tasks are retrieved from a queue and ranked based on predefined priority levels.
- Resource Selection: Once tasks are prioritized, the available containers (resources) are matched to tasks based on their capability to satisfy the task's execution requirements.

MAMTS-DR ensures that high-priority tasks are executed first and mapped to resources with the best trade-off between performance and cost. This technique not only meets deadline constraints but also enhances resource utilization by reducing idle times and avoiding bottlenecks in container clusters.

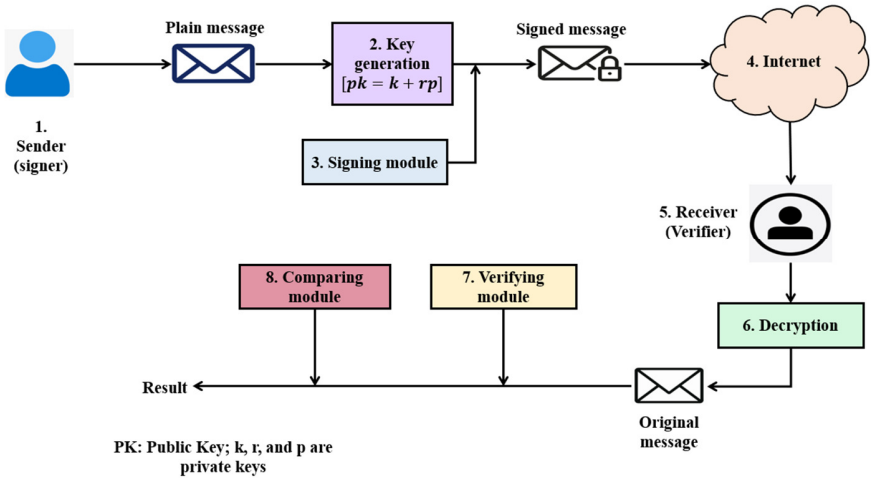

Fig. 1. Architecture of the proposed method.

B. Containers' Encryption Using ECDSA

It is important to ensure that container migration is performed securely through encryption-enabled migration with the help of the blockchain. To illustrate the operational flow of the proposed digital signature scheme, it is essential to understand the underlying architectural model, as shown in Figure 1. Contemporary cryptographic frameworks typically adopt a layered architecture, with each layer fulfilling a distinct security function. The proposed model is structured similarly and comprises key components, including the sender, encryption module, signing module, Internet (communication medium), receiver, decryption module, verification module, and a comparison module.

The proposed digital signature approach uses a private key k and a secret trapdoor p associated with the sender to produce a Digital Signature (DS). The signature consists of two parts, denoted s_1 and s_2, which play vital roles in the signing and verification processes, respectively. In addition, the scheme introduces new foundational formulas to support the signing and verification stages, based on the following primitives:

- A sender integer random r is split into binary random numbers r_1 and r_2.
- A sender's private key is $K_s = (k_s, r_2)$
- The sender'x public key is $PK_s = (pk_s, n, n', r_1)$, where $Pk_s = k_s + r \times p$, $n = p \times q$, and n' denotes a small modulus.
- A plain message m is encrypted through the function Enc using the receiver's public key pk_r, $Enc_{pk_r}(m) = c$.
- A plain message m is signed through a signing function $Sig()$ with the help of the sender's private keys (k_s, r_2, p).
- A receiver's private key is k_r.
- A cipher message c is decrypted through the function Dec by the receiver's private key k_r, $Dec_{k_r}(c) = m$.
- A received signature Sig is verified through the verifying function $Verf()$ by the support of the sender's public keys (pk_s, r_1).

To suggest a public key ($pk = k + r \times p$) in a signature, a significant concept of the proposed approach involves binary phases: primarily, a signature is partitioned into two parts, s_1 and s_2, one utilizing k and another utilizing $r \times p$. Then, r is arbitrarily split into r_1 and r_2, when one is kept secret (r_2) and the other is public, thus, attaining $s_1 = m^k$ and $s_2 = m^{r_2 \times p}$. The receiver can first easily compute $x = s_1 \times s_2^{r_1}$, that is equal to $m^{k+r\times p}$, then estimate $y = m^{kp}$, and then compare them.

1) Signing Procedure

In this procedure, the encryption step is an important part because the resulting ciphertext is later utilized during the verification phase to retrieve the actual message m and confirm data integrity. The sender encrypts the message to initiate the signing process. This encryption step is formally represented as:

$$c = Enc_{pk_r} \quad (1)$$

where c is the ciphertext obtained after encrypting the message m with the receiver's public key. This ensures confidentiality during transmission. Enc_{pk_r} is the encryption function that uses the receiver's public key pk_r. Subsequently, a sender signs a message m by estimating s_1 and s_2, which involves their private keys k_s, r_2, and p as:

$$s_1 = m^{k_s} \, mod \, n' \quad (2)$$

$$s_2 = m^{r_2 \times p} \, mod \, n' \quad (3)$$

where s_1 and s_2 illustrates the first and second components of the digital signature individually, and p is a secret parameter. A digital signature sig_m is sent to the receiver:

$$sig_m = (s_1, s_2) \quad (4)$$

where sig_m is a complete digital signature pair for the message m.

2) Verification Procedure

After the decryption of the cipher text c and recovering the actual message, $m = Dec_{k_r}(c)$, the receiver must estimate a value x through the sender's public key pk_s as:

$$x = m^{pk_s} \, mod \, n' \quad (5)$$

$$y = \left(s_1 \times s_2^{r_1}\right) mod \, n' \quad (6)$$

The comparison process is:

$$x = y \quad (7)$$

where x is an intermediate verification value, derived by raising the message m to the power of the sender's public key pk_s, and y is the other verification value computed from the signature components s_1 and s_2 and the random parameter r_1, respectively.

Thus, if the proposal is right, an information receiver can effortlessly verify sender authenticity through the sender's public key. Algorithm 1 illustrates the pseudocode for the MATS-DR approach for better reproducibility.

```
Algorithm 1. Pseudocode for MATS-DR
Input:
  T = {t1, t2, ..., tn} set of tasks with
  priorities and deadlines
  C = {c1, c2, ..., cm} set of containers with
  available resources
Output:
  Mapping M: T → C
Sort T by (priority descending, deadline
ascending)
For each task t in T do
  R_t ← {c ∈ C | available(c) ≥ demand(t) AND
    finish_time(c) + exec_time(t) ≤ deadline(t)}
  If R_t = ∅ then mark t as delayed and
  continue
  For each c ∈ R_t do
    cost(c,t) = Makespan(c,t) +
      α · EncOverhead(c,t) + β · Energy(c,t)
  End for
  Select c* = argmin cost(c,t)
  Assign t → c*, update c* state
End for
Return M
```

IV. SIMULATION RESULTS

A series of experiments was employed to validate the performance and effectiveness of the proposed container-scheduling framework. Given the dynamic nature of cloud environments, the Container CloudSim simulation platform was selected for implementation purposes.

The experimental workload was derived from the GWA-T-12 BitBrains dataset [29], which includes traces of real-world business applications. This dataset reflects the operational characteristics of 1,750 containers that process 8,260 tasks. For this study, the fast-storage subset of the dataset was used, focusing on the performance data of 1,250 containers. The dataset was stored in .CSV files, each containing detailed logs, such as timestamps, allocated cores, CPU and memory usage, and throughput metrics related to both disk and network performance.

The uniformity of data and tasks used in this research is ensured by controlled selection and preprocessing of the GWA-T-12 BitBrains dataset, which offers standardized traces of real-world business applications over different VMs and containers. This dataset is widely recognized in cloud computing research for its consistent structure, homogeneous performance metrics, and uniform temporal granularity, making it ideal for benchmarking scheduling algorithms. To maintain data uniformity, the experimental workloads are effectively processed to preserve consistent feature distributions such as CPU utilization, memory consumption, and bandwidth across all simulation runs.

The experimental data were split, with 80% used for training and 20% for testing the proposed scheduling model. The splitting was performed randomly at the task level while

ensuring that tasks from the same job or container are not simultaneously placed in both training and testing sets. This guarantees that no task or workload trace instance overlaps between training and testing, thus eliminating the data leakage and conserving the independence of estimation. The core task attributes used for scheduling include the CPU and memory requirements, which are also employed to assess the load level of each container. To maintain the load balance, the utility range of each container is calculated, which helps determine both the utilization levels and available residual resources for task allocation. The system used to implement and test the proposed method was configured with an Intel Core i5 processor, 6 GB of RAM, and a 64-bit operating system.

TABLE I. PARAMETER SETTINGS

Parameters	Values
Number of tasks	5000
Number of hosts	10
Number of containers	50
Number of data-centers	3
Bandwidth	1 GB/s
Storage	1 GB

A range of performance metrics was employed to comprehensively evaluate the effectiveness of the proposed container scheduling strategy. These metrics are critical for analyzing the system efficiency, responsiveness, resource consumption, and overall scheduling quality. The key performance indicators used in this study are as follows.

- CPU utilization measures the extent to which the processor's capacity is efficiently used by the assigned tasks, ensuring that computing resources are not underutilized or idle. This reflects how effectively the CPU handles dynamic task allocation across containers, formulated as:

$$CPU_{util} = \frac{\sum_{i=1}^{w}(p^i \times \omega^i)}{C_P \times M} \quad (8)$$

- Makespan is utilized to identify the overall completion time of the final task leaving the container, mathematically formulated as:

$$M_S = max\ finish\ time\ (w_f) \quad (9)$$

- Energy utilization: The total energy consumed by the containers within the data center is quantified to execute the assigned tasks. This metric is crucial for evaluating the energy efficiency of the scheduling algorithm, particularly in large-scale multi-tenant cloud environments, where minimizing power consumption is a key optimization goal:

$$E_n(t_1, t_2) = \int_{t_1}^{t_2} \sum f(CPU_{util}(t))dt \quad (10)$$

- Execution time describes the amount of time required by a system to start and execute tasks.
- Execution cost is estimated from the price of a container instance, as well as the response time of a system.

For an impartial comparison of the proposed ECDSA method with state-of-the-art methods, this study implemented all cryptographic methods under a similar experimental environment. Existing methods, such as the Advanced Encryption Standard (AES), Data Encryption Standard (DES), Rivest Shamir-Adleman (RSA), Elliptic Curve Cryptography (ECC), and Multi-objective Crow Search Optimization (MOCSO) [23], are compared and estimated using the proposed ECDSA method as follows. To validate the effectiveness of the proposed model, this study compares the results of the proposed method with previous works that used similar datasets and scheduling frameworks.

Table II presents the performance evaluation of encryption and decryption delays based on the number of key sizes. ECDSA consistently delivers the lowest encryption and decryption times, making it the most suitable for real-time cloud operations. RSA has the longest delay and ECDSA outperforms AES, DES, and ECC in terms of both encryption and decryption speeds. The lower computational overhead of ECDSA ensures fast migration and signature processing, which are essential in dynamic cloud environments. This lightweight behavior directly contributes to reduced task waiting time and improved system responsiveness. Hence, ECDSA is optimal for secure and efficient container communication.

Table III shows a performance analysis of energy consumption based on the number of containers. ECDSA showed the lowest energy usage in all scenarios, proving its efficiency and scalability. As the container count increases, other methods show sharp energy increases, particularly RSA and DES. In contrast, ECDSA maintains a controlled and consistent power footprint, which is crucial for sustainable cloud datacenters. Lower energy usage implies not only cost savings but also aligns with green computing objectives. The results confirmed that ECDSA-based scheduling ensures high performance while maintaining minimal energy demands, making it ideal for large-scale multi-cloud deployment.

TABLE II. ESTIMATION OF ENCRYPTION AND DECRYPTION DELAY OF 100 MB FILE (S) BASED ON DIFFERENT KEY SIZES (BITS)

Metrices	Methods	Key size (bits)			
		128	256	384	512
Encryption delay (s)	ES	0.006	0.018	0.029	0.031
	DES	0.012	0.016	0.019	0.023
	RSA	0.034	0.048	0.061	0.085
	ECC	0.009	0.022	0.019	0.028
	ECDSA	0.01	0.013	0.015	0.021
Decryption delay (s)	AES	0.005	0.007	0.008	0.010
	DES	0.011	0.015	0.017	0.021
	RSA	0.029	0.041	0.054	0.072
	ECC	0.008	0.010	0.012	0.016
	ECDSA	0.003	0.005	0.006	0.008

TABLE III. ESTIMATION OF TOTAL ENERGY CONSUMPTION (MW) BASED ON NUMBER OF CONTAINERS

Methods	Number of containers			
	500K	1M	1.5M	2M
AES	17.1	67.8	74.2	109.7
DES	18.6	72.4	78.9	116.2
RSA	21.4	89.2	93.5	131.3
ECC	16.2	63.7	70.1	104.8
ECDSA	15.4	64.3	68.3	101.4

Table IV presents a performance analysis of makespan based on the number of workloads. Makespan was evaluated as the total time required to complete all tasks for the different encryption methods under varying workload sizes. ECDSA consistently achieved the shortest makespan, even as the number of workloads increased. Unlike RSA and DES, which exhibit significant increases, ECDSA maintains steady performance with a minimal increase in completion time. This confirms the ability of the proposed method to handle a large number of tasks efficiently. A low makespan reflects better container scheduling and faster turnaround times, which are essential in dynamic and real-time cloud environments. Thus, ECDSA contributes to faster service delivery with fewer delays.

TABLE IV. OERFORMANCE ANALYSIS OF THE MAKESPAN (S) WITH RESPECT TO THE NUMBER OF WORKLOADS.

Methods	Number of Workloads			
	1000	2000	3000	5000
AES	108.3	121.2	136.5	159.3
DES	115.2	129.6	143.7	169.8
RSA	123.7	138.3	149.1	175.2
ECC	95.4	104.8	115.2	138.6
MOCSO [23]	94.58	95.37	101.82	130.21
ECDSA	91.4	93.5	98.4	127.4

Table V presents a performance analysis of the execution time and execution cost based on the number of workloads. As workload increases, ECDSA maintains the lowest execution time and cost, showing excellent performance efficiency. This advantage is crucial in the IaaS and SaaS models, where cost and delay minimization are the top priorities. The marginal increase in cost with workload confirms cost scalability, whereas a reduced execution time ensures faster processing. Compared to traditional algorithms, ECDSA reduces overhead while maintaining strong security, making it ideal for modern multi-tenant cloud environments that require economic and rapid task handling.

TABLE V. PERFORMANCE ANALYSIS OF EXECUTION TIME (MS) AND EXECUTION COST (C$) WITH RESPECT TO THE NUMBER OF WORKLOADS.

Metrics	Methods	Workload			
		1000	2000	3000	5000
Execution time	AES	31.3	34.5	38.4	41.3
	DES	29.3	32.4	35.3	39.4
	RSA	25.4	27.3	28.4	36.6
	ECC	21.3	23.4	24.4	33.4
	MOCSO [23]	21.7	22.51	24.86	33.18
	ECDSA	19.3	20.3	22.5	31.4
Execution cost	AES	29.6	31.8	29.4	52.4
	DES	26.2	29.4	27.3	51.3
	RSA	21.3	27.3	24.5	48.2
	ECC	19.3	25.3	22.4	46.5
	MOCSO [23]	20.77	25.28	26.54	47.92
	ECDSA	18.3	23.38	21.3	44.2

Table VI presents the statistical validation of MAMTS-DR against existing methods using variance analysis, t-test, and ANOVA. These statistical measures were computed for three key performance parameters, makespan, energy consumption, and execution time, because they represent the most critical indicators of scheduling efficiency and resource optimization in multi-cloud environments. The variance values illustrate that MAMTS-DR attains minimum variability over repeated runs, indicating consistent performance. The t-test and ANOVA results further confirm that the improvements in makespan, energy consumption, and execution time are statistically significant, with p-values below 0.05. These findings demonstrate that the observed gains are not due to random variations, but reflect genuine improvements. By combining variance stability with significance testing, the results reinforce the reliability of the proposed approach. In general, this analysis strengthens the credibility of performance claims across different workloads. The statistical tests are based on the null hypothesis (H0) that there is no significant difference in the mean values of makespan, energy consumption, and execution time between MAMTS-DR and existing methods, and an alternative hypothesis (H1) that a significant difference exists. Since all p-values are below 0.05, H_0 is rejected, confirming that the observed improvements are statistically significant and not due to random variation.

TABLE VI. STATISTICAL ANALYSIS OF PROPOSED METHOD USING GWA-T-12 BITBRAINS DATASET

Methods	Variance	t-test	ANOVA (p-value)
AES	0.0042	0.051	0.049
DES	0.038	0.048	0.047
RSA	0.051	0.046	0.045
ECC	0.034	0.041	0.040
ECDSA	0.021	0.032	0.031

Figure 2 illustrates the graphical representation of resource utilization across different workloads. The results show that the proposed MAMTS-DR consistently achieves higher utilization compared to existing methods, indicating efficient mapping of tasks to available resources. As workload increases, other methods display fluctuations and underutilization, whereas MAMTS-DR maintains stability and minimizes idle resources.

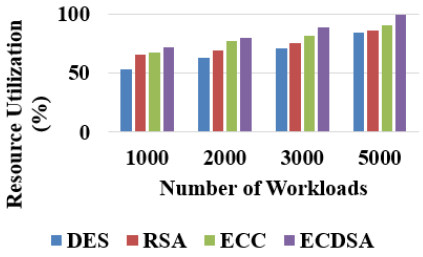

Fig. 2. Graphical representation of resource utilization based on the number of workloads.

Figure 3 presents the Service Level Agreement Violation (SLAV) with respect to increasing workloads. The proposed MAMTS-DR shows consistently lower SLAV values compared to baseline methods, reflecting its ability to maintain a balanced workload distribution across containers. Lower SLAV indicates reduced imbalance, fewer bottlenecks, and better utilization of multi-cloud resources.

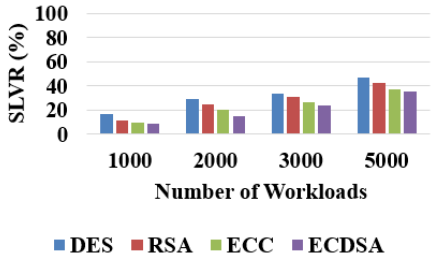

Fig. 3. Graphical representation of SLAV based on the number of workloads.

A. Discussion

The simulation results validate the robustness and efficiency of the proposed container scheduling model. Among the cryptographic schemes tested, ECDSA consistently outperformed the others in terms of encryption/decryption speed and energy consumption. Priority-aware task scheduling using MAMTS-DR contributed to lower makespan values, even as the workload increased. The proposed method maintained a consistent execution cost as the number of containers and task volume increased, demonstrating its cost efficiency and scalability. Unlike RSA and DES, which incur significant computational overhead, ECDSA demonstrates excellent responsiveness, especially in large task environments. This architectural design supports secure container migration while avoiding excessive encryption delays, which is a common limitation in legacy systems. Energy usage across containers was also significantly lower for ECDSA, aligning with sustainability goals in data center operations. Moreover, the extended DAG-based DCMM-MTS strategy ensures optimal task-to-container mapping under deadline constraints, contributing to enhanced resource utilization and reduced load imbalance. These observations demonstrate that the proposed architecture can be reliably adopted for secure, scalable, and energy-efficient container orchestration in real-time multicloud environments.

V. CONCLUSION

This study introduced a two-stage container scheduling approach that efficiently deploys containers and addresses imbalances across nodes through migration. The proposed solution expresses a scheduling procedure as an optimization problem and incorporates different objective functions and restrictions to improve server consolidation and reduce energy consumption. Encryption was used to ensure the confidentiality of containers during migration. The encryption and decryption costs are included in the optimization constraints of the proposed solution. This solution is effective in multi-data center cloud environments, contributing to the enhancement of container composition systems in real-world scenarios. The experimental scale is intentionally constrained because of the computational and time constraints of the simulation environment. Running large-scale multi-cloud simulations with thousands of tasks and containers requires high-performance infrastructure and extended simulation time. Thus, a smaller but representative configuration (10 hosts and 50 containers) was used to validate feasibility and correctness, while large-scale scalability testing is planned as a separate extended study.

Future work will focus on developing a hybrid optimization algorithm and estimating the significance of the proposed method using a broader range of performance metrics to enhance the overall results. Moreover, although the experiments conducted with 10 hosts and 50 containers demonstrate the feasibility and effectiveness of the proposed model, this scale is smaller than real-world multi-cloud deployments. Therefore, while the results confirm efficiency and secure scheduling in controlled settings, large-scale validation will be pursued to further substantiate the scalability claims.

ACKNOWLEDGMENT

The authors express their sincere gratitude to BMS Institute of Technology and Management, Bengaluru, M.S. Ramaiah Institute of Technology, Bengaluru, and Sai Vidya Institute of Technology, Bengaluru, for their constant support and encouragement during the course of this research work. We also extend our thanks to our colleagues and peers for their valuable suggestions and guidance.

REFERENCES

[1] A. I. Abueid, "Big Data and Cloud Computing Opportunities and Application Areas," *Engineering, Technology & Applied Science Research*, vol. 14, no. 3, pp. 14509–14516, June 2024, https://doi.org/10.48084/etasr.7339.

[2] K. Geeta and V. K. Prasad, "Multi-objective cloud load-balancing with hybrid optimization," *International Journal of Computers and Applications*, vol. 45, no. 10, pp. 611–625, Oct. 2023, https://doi.org/10.1080/1206212X.2023.2260616.

[3] A. Y. Hamed, M. K. Elnahary, F. S. Alsubaei, and h. H. El-Sayed, "Optimization Task Scheduling Using Cooperation Search Algorithm for Heterogeneous Cloud Computing Systems," *Computers, Materials and Continua*, vol. 74, no. 1, pp. 2133–2148, Aug. 2022, https://doi.org/10.32604/cmc.2023.032215.

[4] M. Mahdizadeh, A. Montazerolghaem, and K. Jamshidi, "Task scheduling and load balancing in SDN-based cloud computing: A review of relevant research," *Journal of Engineering Research*, Nov. 2024, https://doi.org/10.1016/j.jer.2024.11.002.

[5] E. M. Elshahed, R. M. Abdelmoneem, E. Shaaban, H. A. Elzahed, and S. M. Al-Tabbakh, "Prioritized scheduling technique for healthcare tasks in cloud computing," *The Journal of Supercomputing*, vol. 79, no. 5, pp. 4895–4916, Mar. 2023, https://doi.org/10.1007/s11227-022-04823-7.

[6] S. Mangalampalli, G. R. Karri, M. Kumar, O. I. Khalaf, C. A. T. Romero, and G. A. Sahib, "DRLBTSA: Deep reinforcement learning based task-scheduling algorithm in cloud computing," *Multimedia Tools and Applications*, vol. 83, no. 3, pp. 8359–8387, Jan. 2024, https://doi.org/10.1007/s11042-023-16008-2.

[7] K. V. Kumar and A. Rajesh, "Multi-Objective Load Balancing in Cloud Computing: A Meta-Heuristic Approach," *Cybernetics and Systems*, vol. 54, no. 8, pp. 1466–1493, Nov. 2023, https://doi.org/10.1080/01969722.2022.2145656.

[8] S. Iftikhar *et al.*, "HunterPlus: AI based energy-efficient task scheduling for cloud–fog computing environments," *Internet of Things*, vol. 21, Apr. 2023, Art. no. 100667, https://doi.org/10.1016/j.iot.2022.100667.

[9] X. Zhang, "A fine-grained task scheduling mechanism for digital economy services based on intelligent edge and cloud computing," *Journal of Cloud Computing*, vol. 12, no. 1, Mar. 2023, Art. no. 30, https://doi.org/10.1186/s13677-023-00402-0.

[10] J. Pan, Y. Wei, L. Meng, and X. Meng, "A dual scheduling framework for task and resource allocation in clouds using deep reinforcement learning," *Journal of King Saud University Computer and Information Sciences*, vol. 37, no. 5, June 2025, Art. no. 81, https://doi.org/10.1007/s44443-025-00092-5.

[11] T. Singh and A. Kumar, "Analyzing Security and Privacy issues for Multi-Cloud Service Providers Using Nessus," in *2023 Fifth International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, Erode, India, Feb. 2023, pp. 01–08, https://doi.org/10.1109/ICECCT56650.2023.10179727.

[12] P. K. Mishra and A. K. Chaturvedi, "An Improved Laxity based Cost Efficient Task Scheduling Approach for Cloud-Fog Environment," *Engineering, Technology & Applied Science Research*, vol. 15, no. 1, pp. 19037–19044, Feb. 2025, https://doi.org/10.48084/etasr.8595.

[13] G. Sreelatha, C. K. K. Reddy, M. M. Hanafiah, and R. Madana Mohana, "Hybrid Electro search beetle optimization based task scheduling and game theory SOA based resource allocation in multi cloud computing," *Software: Practice and Experience*, vol. 55, no. 2, pp. 307–331, 2025, https://doi.org/10.1002/spe.3370.

[14] S. Sharma and P. S. Rawat, "Efficient resource allocation in cloud environment using SHO-ANN-based hybrid approach," *Sustainable Operations and Computers*, vol. 5, pp. 141–155, Jan. 2024, https://doi.org/10.1016/j.susoc.2024.07.001.

[15] Y. Liang, G. Xu, H. Shen, N. Ruan, and Y. Wang, "Towards Efficient Job Scheduling for Cumulative Data Processing in Multi-Cloud Environments," *Electronics*, vol. 14, no. 7, Jan. 2025, Art. no. 1332, https://doi.org/10.3390/electronics14071332.

[16] A. Nelli and R. Johdand, "A Min-Max Workload Scheduling Technique Using Soft-Computing Approach in Multi-Cloud Platform," *International Journal of Intelligent Engineering and Systems*, vol. 16, no. 4, pp. 115–124, Aug. 2023, https://doi.org/10.22266/ijies2023.0831.10.

[17] J. Zhou *et al.*, "Comparative analysis of metaheuristic load balancing algorithms for efficient load balancing in cloud computing," *Journal of Cloud Computing*, vol. 12, no. 1, June 2023, Art. no. 85, https://doi.org/10.1186/s13677-023-00453-3.

[18] J. Dogani, A. Yazdanpanah, A. Zare, and F. Khunjush, "A two-tier multi-objective service placement in container-based fog-cloud computing platforms," *Cluster Computing*, vol. 27, no. 4, pp. 4491–4514, July 2024, https://doi.org/10.1007/s10586-023-04183-8.

[19] S. Badri *et al.*, "An Efficient and Secure Model Using Adaptive Optimal Deep Learning for Task Scheduling in Cloud Computing," *Electronics*, vol. 12, no. 6, Jan. 2023, Art. no. 1441, https://doi.org/10.3390/electronics12061441.

[20] A. M. Alhassan, "Secure multi-cloud resource allocation with SDN and self-adaptive authentication," *Ain Shams Engineering Journal*, vol. 15, no. 6, June 2024, Art. no. 102742, https://doi.org/10.1016/j.asej.2024.102742.

[21] M. A. Altahat, T. Daradkeh, and A. Agarwal, "Optimized Encryption-Integrated Strategy for Containers Scheduling and Secure Migration in Multi-Cloud Data Centers," *IEEE Access*, vol. 12, pp. 51330–51345, 2024, https://doi.org/10.1109/ACCESS.2024.3386169.

[22] M. A. Altahat, T. Daradkeh, and A. Agarwal, "Virtual machine scheduling and migration management across multi-cloud data centers: blockchain-based versus centralized frameworks," *Journal of Cloud Computing*, vol. 14, no. 1, Jan. 2025, Art. no. 1, https://doi.org/10.1186/s13677-024-00724-7.

[23] M. A. Naji Saif, S. Niranjan Aradhya, B. A. Hezam Murshed, O. A. M. Farhan Alnaggar, and I. M. S. Ali, "Multi-objective container scheduling and multi-path routing for elastic business process management in autonomic multi-tenant cloud," *Concurrency and Computation: Practice and Experience*, vol. 35, no. 6, 2023, Art. no. e7584, https://doi.org/10.1002/cpe.7584.

[24] S. S. Sefati, A. M. Nor, B. Arasteh, R. Craciunescu, and C. R. Comsa, "A Probabilistic Approach to Load Balancing in Multi-Cloud Environments via Machine Learning and Optimization Algorithms," *Journal of Grid Computing*, vol. 23, no. 2, Apr. 2025, Art. no. 16, https://doi.org/10.1007/s10723-025-09805-6.

[25] M. A. N. Saif, S. K. Niranjan, B. A. H. Murshed, F. A. Ghanem, and A. A. Q. Ahmed, "CSO-ILB: chicken swarm optimized inter-cloud load balancer for elastic containerized multi-cloud environment," *The Journal of Supercomputing*, vol. 79, no. 1, pp. 1111–1155, Jan. 2023, https://doi.org/10.1007/s11227-022-04688-w.

[26] N. Elsakaan and K. Amroun, "A novel multi-level hybrid load balancing and tasks scheduling algorithm for cloud computing environment," *The Journal of Supercomputing*, vol. 80, no. 9, pp. 13434–13474, June 2024, https://doi.org/10.1007/s11227-024-05990-5.

[27] S. Fugkeaw, S. Rattagool, P. Jiangthiranan, and P. Pholwiset, "FPRESSO: Fast and Privacy-Preserving SSO Authentication With Dynamic Load Balancing for Multi-Cloud-Based Web Applications," *IEEE Access*, vol. 12, pp. 157888–157900, 2024, https://doi.org/10.1109/ACCESS.2024.3485996.

[28] P. Suresh *et al.*, "Optimized task scheduling approach with fault tolerant load balancing using multi-objective cat swarm optimization for multi-cloud environment," *Applied Soft Computing*, vol. 165, Nov. 2024, Art. no. 112129, https://doi.org/10.1016/j.asoc.2024.112129.

[29] "gwa-bitbrains." Kaggle, [Online]. Available: https://www.kaggle.com/datasets/gauravdhamane/gwa-bitbrains.

AUTHORS PROFILE

Komala Rangappa is an Assistant Professor at the Department of MCA, Ramaiah Institute of Technology, Bengaluru. She has published nearly 10 research papers in international journals and conferences. She has attended more than 30 FDPs and has filed and published 2 patents.

Arun Kumar Banavara Ramaswamy is a distinguished Professor in Information Science and Engineering, renowned for his extensive contributions to research with a profound interest in the field of Cyber Privacy & Security, Blockchain Technology, and Cloud /Network Computing. Leveraging extensive expertise, his work profoundly advances cyber defense principles, shaping the development of resilient and secure information systems. He is the author of two books, 15 book chapters, and more than 100 publications in international and indexed conferences/journals.

Shreyas Arun Kumar is a Computer Science and Engineering student at Sai Vidya Institute of Technology (affiliated with VTU) with a profound interest in the field of cybersecurity. He is actively engaged in studying and developing solutions to modern digital threats. His work is driven by a dedication to advancing the principles of cyber defense and contributing to the development of more secure information systems.