A
Mini Project Report
On

"A Security Deduplicated Cloud Storage with Encrypted Two-Party Interactions in Cyber-Physical Systems"

Submitted to JNTUH in partial fulfilment of the
Requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

By

MOHAMMAD ABDUL RAHMAN **21N61A05A0**

Under the Guidance of
Mrs. ZAHRA TARANNUM
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VIVEKANANDA INSTITUTE OF TECHNOLOGY & SCIENCE
(Approved by AICTE New Delhi & Affiliated to JNTU, Hyderabad) **An ISO 9001:2015 Certified Institution**
KARIMNAGAR-505501

2021-2025

VIVEKANANDA INSTITUTE OF TECHNOLOGY & SCIENCE(N6)

(Approved by AICTE New Delhi & Affiliated to JNTU, Hyderabad) **An ISO 9001:2015 Certified Institution**

KARIMNAGAR-505001

CERTIFICATE

This is to certify that the mini-project report titled **A Security Deduplicated Cloud Storage with Encrypted Two-Party Interactions in Cyber-Physical Systems** is being submitted by **MOHAMMAD ABDUL RAHMAN 21N61A05A0** in B. Tech IV-I semester, Computer Science & Engineering is a record bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.

Internal Guide

Mrs. ZAHRA TARANNUM
Assistant Professor

Head of the Department

Dr.M.V.Hanumanthareddy
Dept of CSE

External Examiner

Principal

Dr.M.V.Hanumanthareddy

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our guide, **Mrs. ZAHRA TARANNUM** Assistant Professor whose knowledge and guidance has motivated us to achieve goals we never thought possible. She has consistently been a source of motivation, encouragement, and inspiration. The time we have spent working under her supervision has truly been a pleasure.

We thank our H.O.D **Dr.M.V.Hanumanthareddy** for his effort and indefatigable guidance rendered throughout the progress of project work. Thanks to programmers and nonteaching staff of CSE Department of VITS(N6).

We thank our Principal **Dr.M.V.Hanumanthareddy** and Management for providing excellent facilities to carry out project work.

Finally, Special thanks to our parents for their support and encouragement throughout this course. And thanks to our friends and well wishers for their constant support.

MOHAMMAD ABDUL RAHMAN
21N61A05A0

ABSTRACT

Cloud envisioned Cyber-Physical Systems (CCPS) is a practical technology that relies on the interaction among cyber elements like mobile users to transfer data in cloud computing. In CCPS, cloud storage applies data deduplication techniques aiming to save data storage and bandwidth for real-time services. In this infrastructure, data deduplication eliminates duplicate data to increase the performance of the CCPS application. However, it incurs security threats and privacy risks. For example, the encryption from independent users with different keys is not compatible with data deduplication. In this area, several types of research have been done. Nevertheless, they are suffering from a lack of security, high performance, and applicability. Motivated by this, we propose a message Lock Encryption with never-decrypt homomorphic Encryption (LEVER) protocol between the uploading CCPS user and cloud storage to reconcile the encryption and data deduplication. Interestingly, LEVER is the first brute-force resilient encrypted deduplication with only cryptographic two-party interactions. We perform several numerical analysis of LEVER and confirm that it provides high performance and practicality compared to the literature.

LEVER is a novel framework designed to address the growing challenges of secure and efficient cloud storage in the context of Cyber-Physical Systems (CPS). As CPS increasingly rely on data generated by interconnected physical devices such as sensors, actuators, and IoT devices, ensuring the confidentiality, integrity, and availability of data while optimizing storage efficiency becomes a critical concern. LEVER integrates advanced cryptographic techniques, secure two-party communication protocols, and data deduplication mechanisms to provide a robust solution for cloud storage.

At the core of LEVER is its ability to perform secure deduplication, which minimizes redundant data storage without compromising data privacy. The system employs encryption strategies to protect data during interactions between two parties (e.g., user and cloud provider), ensuring that sensitive information remains confidential even in the presence of untrusted intermediaries. LEVER's approach leverages secure multi-party computation and homomorphic encryption, enabling computations on encrypted data while maintaining privacy.

LEVER is particularly suited for environments that demand real-time data synchronization and storage, such as industrial automation, healthcare, smart cities, and transportation systems. By integrating LEVER into CPS, organizations can enhance data storage efficiency, reduce operational costs, and ensure compliance with stringent security and privacy regulations. This paper discusses the system's architecture, cryptographic protocols, deduplication techniques, and integration with CPS, highlighting its potential to transform how cloud storage is managed and secured in modern cyber-physical environments.

Table of Contents

1. INTRODUCTION

1.1 Overview of LEVER

LEVER is an innovative framework designed to offer **secure, deduplicated cloud storage** with a focus on **encrypted two-party interactions** in **Cyber-Physical Systems (CPS)**. As the proliferation of Internet of Things (IoT) devices and sensors continues to drive the generation of vast amounts of data in real-time, LEVER provides a solution that addresses both the **security concerns** of storing sensitive data in the cloud and the **storage inefficiency** of handling large volumes of potentially redundant data.

Secure Cloud Storage: LEVER ensures that data stored in the cloud is encrypted, maintaining its confidentiality and integrity, even when transmitted over untrusted networks. By employing advanced cryptographic techniques, LEVER ensures that only authorized parties can access and manipulate the data.

Efficient Data Deduplication: LEVER introduces an efficient deduplication mechanism that helps reduce the amount of stored data without compromising the privacy of the users. Deduplication helps in identifying and eliminating duplicate copies of data, thus minimizing storage costs while improving system efficiency.

Two-Party Secure Communication: One of the unique features of LEVER is its ability to support encrypted interactions between two parties—typically a user (data owner) and the cloud provider—using secure cryptographic protocols. These protocols ensure that data remains encrypted during transfer and while being processed, preventing any exposure to unauthorized parties.

Seamless Integration with Cyber-Physical Systems (CPS): LEVER is specifically designed to work within the context of CPS, which involve a combination of physical systems (e.g., IoT devices, sensors, actuators) and computational elements. LEVER facilitates real-time data collection, processing, and secure storage of data in environments where latency, security, and data volume are critical factors.

Cloud Storage Layer: The cloud storage layer is where data is securely stored in an encrypted format. This layer interacts with the cryptographic modules to ensure that all data stored in the cloud is protected against unauthorized access.

Cryptographic Modules: At the heart of LEVER's security framework are cryptographic modules that ensure data confidentiality, integrity, and authenticity. LEVER employs both symmetric and asymmetric encryption, digital signatures, and homomorphic encryption to facilitate secure data storage and processing. Homomorphic encryption allows computations on encrypted data without revealing the underlying plaintext, ensuring privacy even in the cloud environment.

Deduplication Engine: The deduplication engine plays a key role in reducing storage requirements. By comparing data blocks and detecting identical or redundant data, it eliminates unnecessary copies without compromising the confidentiality of the information. The challenge of deduplicating encrypted data is addressed using specialized techniques that allow for deduplication without decrypting the data, preserving privacy.

Two-Party Interaction Protocols: LEVER supports two-party secure communication using cryptographic protocols designed to enable data exchange between the user and the cloud provider in a privacy-preserving manner. These protocols ensure that neither party can view the other's private data unless explicitly authorized.

Cyber-Physical Systems Integration: LEVER's design takes into account the unique challenges of CPS, such as the real-time nature of data collection, low-latency requirements, and the need for large-scale, distributed storage systems. By supporting efficient synchronization between devices and cloud storage, LEVER helps ensure that the data collected from sensors and devices is processed and stored securely in real time.

Data Privacy: By utilizing strong encryption and secure communication protocols, LEVER ensures that sensitive data remains private and protected from unauthorized access, even in the cloud environment.

Storage Efficiency: Through its deduplication mechanism, LEVER significantly reduces the storage footprint, helping users save on cloud storage costs while maintaining security.

Scalability: LEVER is designed to scale in distributed cloud environments, making it suitable for large-scale CPS applications with numerous connected devices and vast amounts of data.

Low Latency: LEVER is optimized for low-latency data interactions, which is critical in real-time systems like smart cities, healthcare, and industrial automation, where timely data access and decision-making are essential.

Seamless CPS Integration: LEVER's ability to seamlessly integrate with CPS ensures that the data generated by physical systems can be securely stored, accessed, and analyzed in the cloud without compromising performance or security.

Smart Cities: In a smart city environment, IoT devices such as traffic sensors, environmental monitors, and smart meters generate large volumes of data. LEVER can help securely store and deduplicate this data, providing a cost-effective solution for managing smart city infrastructure.

Industrial Automation: In industrial environments, where sensors and actuators control manufacturing processes, LEVER can securely store real-time operational data and ensure that duplicate data from sensors is avoided, leading to more efficient use of storage.

Healthcare: In healthcare, where patient data is highly sensitive and must be protected, LEVER can ensure that medical records and sensor data are securely encrypted and stored in the cloud, with deduplication improving storage efficiency.

Autonomous Vehicles: Autonomous vehicles generate large amounts of data from sensors and cameras in real-time. LEVER can help store this data securely in the cloud while ensuring that only unique data is retained to optimize storage and reduce costs.

1.2 Motivation for the System

Cyber-Physical Systems (CPS) providing data-processing services among the users that can be connected on the Internet. This system is integrating computation, networking, and physical activities to increase the performance of the services. In the CPS, the users can transfer data among themselves using the Internet that emerging Internet of things (IoT) [1]. Thanks to developing cloud computing to enhance the quality of services and increase data transfer volume in the network, CPS and IoT systems can increase their throughput among the users [2]. Such data requires saving in storage to be used for future services/requests. One prominent and efficient technique to save data storage and bandwidth in the cloud Storage is data deduplication. To be precise, data deduplication is a technique storing a single copy of data in the storage. Cloud storage providers (e.g., Dropbox) perform data deduplication to save significant storage.

The increasing reliance on **Cyber-Physical Systems (CPS)**, driven by advancements in **Internet of Things (IoT)** technologies, has led to an explosion in data generation across various domains such as healthcare, industrial automation, smart cities, and transportation. While this influx of data presents significant opportunities for innovation, it also creates a range of **security, privacy, and storage efficiency challenges** that need to be addressed. LEVER was developed to meet these challenges by providing a **secure, efficient, and scalable cloud storage solution** that can handle the unique needs of CPS.

Data deduplication can be done on two sides: *server-side* and *client-side* [1]. In the former, the client first sends the data to the cloud, and then the cloud decides about data deduplication. In later, it takes place on the user side. In this way, the client first sends a request to the cloud. Then, the cloud will check the requested data. If the file exists, it sends back the positive response and applies data deduplication. Otherwise, it returns a negative response. Client-side data deduplication could significantly reduce bandwidth requirements. Hence, cloud storage service providers are willing to use this technique for storing data. Client-side data deduplication can further have the same level of network bandwidth savings via *duplicate check*. Moreover, data deduplication can be exerted at the level of chunks or files. As a result, from the deduplication opportunity point of view, cross-user client-side chunk-level deduplication acts as the most aggressive technique in eliminating redundant

transmissions and storage. This technique will help the CPS to enhance data coordination and system performance [3]. Throughout the paper, *data deduplication* refers to cross-user client-side chunk-level data deduplication in CPS like [4]–[6], unless stated otherwise.

Security and Privacy Concerns. Unfortunately, data deduplication benefits come with the security and privacy threats from an insider (e.g., cloud storage provider) and even the external attacker. For example, *poison attack* [7] aims to achieve the inconsistency between the stored chunk f^0 and the tag $h(f)$ such that the subsequent users with f will have a falsified copy on the cloud after the duplicate check. In particular, the external attacker can poison f by sending tag $h(f)$ but uploading f^0 such that the cloud associates all the subsequent users with tag $h(f)$ to f^0 because of the tag matching. In the end, the user supposed to receive f from the cloud will obtain the falsified chunk, f^0. *Side channel* [8] in the deduplicated storage raises the privacy concern. In particular, the external attacker may gain the existence status of f from the tag response. Due to the low min-entropy nature of the contents in the real world [9], this extra information enables the attacker to adopt the brute-force-like strategy and uncover the sensitive content [8].

Besides, even if message-locked encryption (MLE) [7] is employed, the external attacker will still be able to uncover the user's private data by brute-forcing the low min-entropy contents. Although encrypting the data with a random key, rather than the message-dependent key, is a plausible solution before uploading. However, the encryptions of the identical file from independent users result in different ciphertexts. It leads to losing the storage and bandwidth advantages of data deduplication. In particular, the session key established by the secure connection (e.g., HTTPS) between the user and cloud could be used as an extra layer to encrypt the message locked ciphertext, eliminating the possibility of the brute-force search of content chunks. Nonetheless, various SSL/TLS vulnerabilities have been found. A massive number of HTTPS services are vulnerable to trivial man-in-the-middle attacks, making even the external attacker able to remove the extra encryption layer and derive message-locked ciphertext [2]. The situation becomes more severe because the cloud may also be honest-but-curious, attempting to breach the user's data privacy. A

naïve solution is still the encryption; however, similarly, the improper use of encryption either destroys the deduplication gain or does not have the resilience to the bruteforce attack.

A. Design Challenge

One may have the hybrid use of various solutions to simultaneously develop the deduplicated storage resistant to security and privacy issues. Nevertheless, the particular design of certain encrypted deduplication hinders the joint use with the current brute-force defences. On the other hand, even if the hybrid construction is possible, as each solution for the individual issue has the performance and security weaknesses, the deduplicated storage system based on these building blocks also inherits the performance and security/privacy problems.

There are several protocols in the literature targeting the encrypted data deduplication problems in cloud storage. However, the current encrypted data deduplication either suffer from a brute-force attack or rely on computation-intensive operations and independent key servers. The state-of-the-art method [11] claims to eliminate the need for the independent server but uses cloud users as key servers. The method in [11] also uses two heavyweight cryptographic techniques, password authenticated key exchange (PAKE) and homomorphic encryption (HE), leading to the computation inefficiency.

In this work, we aim to tackle the brute-force attack in cloud storage, applying in CPS. To fill the above gaps and refer to the paper's contribution, some questions arise: i) Is it possible to design an efficient encrypted data deduplication algorithm in CPS? ii) Can we assure that the proposed algorithm could bring data privacy, high/practical performance compared to the literature? And, iii) How can the encrypted method support two-party interaction between the uploader and the cloud server?

B. Design Goals

Here, we list our design goals for the cloud deduplicated storage applied in CPS and the overhead savings, to ensure our design objective.

TABLE I: Comparisons between different encrypted data deduplication methods where ◦ := the method does not support the property, and •:= the method supports the property.

Brute-Force Resiliency. Real applications usually create low min-entropy (i.e., predictable) content *f*, which makes the *brute-force* strategy easy to find out sensitive information encoded by the current deduplication-friendly encryptions. As a result, brute-force resiliency, even for predictable contents, is a desirable goal.

Two Party Interaction. The use of independent trusted servers (or any trusted component) may simplify the system design. Nonetheless, the most critical weakness of independent servers is the impracticality of running them without business justification [11]. Hence, the uploading and downloading in the superior system design involves only the interactions between the uploading user and cloud.

Computation Efficiency. Using only lightweight cryptographic primitives or using heavyweight cryptographic primitives as few times as possible may keep computation efficiency, preventing the severe impact on the deduplication granularity and benefit due to performance degradation.

C. Our contributions

The contributions of this paper are summarized as follows.

- An obstacle in designing encrypted data deduplication is how the cloud can find that *two* distinct encrypted chunks are from the same content. Thus, our first contribution is developing a message Lock Encryption with neVer-decrypt homomorphic EncRyption (LEVER) by taking advantage of the property of homomorphic encryption without further decryption and resiliency against brute-force by external attackers.
- In LEVER, only the uploader and cloud participate in the uploading process, in contrast to most current solutions in need of the third party's participation in CPS.
- LEVER can work transparently with any current cloud storage linked with the CPS users without the need for cloud storage provider's engineering work at the backend.
- We validate LEVER in terms of communication and deduplication costs using *two* datasets, namely Enron [12] and Oxford [13].
- Lastly, in addition to the analysis and numerical simulations, we have a LEVER prototype to demonstrate the practicality. A comparison among different systems is shown in Table I.

The python implementation of LEVER is available in [14].

D. Organization

We organize the paper as follows. Section II describes the related work and highlights the weakness of them compared to LEVER. In Section III, we explain the general setting applied in the paper and the security model. Section IV presents the LEVER, the proposed method. We present

partially homomorphic encryption in V. The performance evaluation includes the results that are described in Section VI. Finally, we conclude the work in Section VII. Table II summarizes the main notations of this paper.

In this part, we explain the solutions applied to encrypted data deduplication.

Convergent Encryption (CE) and its generalization, MLE [7], [15], [18] are the easiest methods to address the privacy issues without compromising the deduplication effectiveness. Specifically, the user calculates and uploads $E_{h(f)}(f)$, where $E_k(\cdot)$ stands for the symmetric encryption with key k. Since the users with f are all can derive the same $h(f)$ and $E_{h(f)}(f)$, the deduplication still takes place on $E_{h(f)}(f)$. The follow-up studies [19], [20] further validate the MLE protocol to preserve message correlation and parameter dependency. This method is weak against the bruteforce attack, particularly in the case of predictable data. The brute-force mainly incurs when we face low min-entropy characteristics, and the keyspace in CE is identical to the plaintext space, which is happened in [19], [20].

DupLESS [9] is a suitable solution to defend against the brute-force attack in cloud storage. In the DupLESS, they introduce an additional key server (KS) to generate the key. In particular, the user u with the aid of KS calculate a content-dependent key k_f for the file f such that no one can drive the key except the user. Later on, deduplication $E_{k_f}(f)$ can be done using k_f. Based on the trusted server, many other approaches are also proposed [16], [21], [22]. Encrypt-with-Signature (EwS) [17] is another solution using KS for calculating $E_{k_f}(f)$. Both of [9], [17] are requiring to communicate with the trusted server to calculate deduplication $E_{k_f}(f)$. It increases the privacy issue in the network.

Liu et al. [11] propose a client-as-a-key-server (CaS) framework, where users are potential key servers, to deduplicate encrypted data. The solutions mentioned above suffer from performance degradation, because of heavyweight cryptographic primitives (e.g., oblivious pseudorandom function [23], homomorphic encryption [24], and PAKE [25]) are used. Also, they are not practically useful because the trusted server does not have a reasonable business justification.

In this section, we describe the considered cross-user Cloud envision CPS and data deduplication architectures (sections III-A and III-B). Then, we present the LEVER's setting (section III-C) and our security models which include storage and threat models (section III-D).

A. The Considered Architectures

Fig. 1a presents the proposed cross-user cloud envision cyber-physical system (CCCPS) architecture. As we can see, user 1 from CPS 1 and user 2 from CPS 2 upload the same file f into the CDD (see the continuous arrows in Fig. 1a). Meanwhile, an adversary aims to gain information about the file f and get access to the encrypted data to change its integrity in CDD (see the logical dashed links in Fig. 1a). We require to impose our cloud server to preserve the data's security and privacy while satisfying the data deduplication technique in the system.

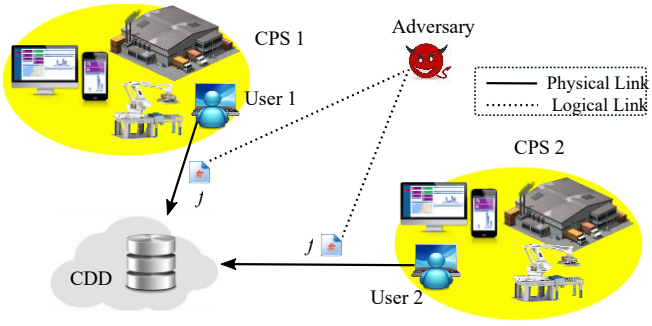
B. Data Deduplication Architecture

Fig. 1b depicted the process of data deduplication in this paper. We consider cloud storage (C), which applies cross-user client-side fixed-sized chunk-level data deduplication. We have a user (u) who performs the *duplicate check* on a chunk by exchanging messages with the cloud. The process of *duplicate check* is as follows, when a user wants to upload a file f (i.e., chunk), he first uses $h(f)$ (i.e., it is a cryptographic hash function like SHA256) to calculate the hash (i.e., also called *tag* of the file f). Then, he sends it to the cloud (e.g., "Is $h(f)$ in the cloud?"). If the cloud finds a copy of $h(f)$ in the memory, he will receive a positive *tag response* and does not require to upload f again. Otherwise, the user will receive a negative tag response and uploads f, and the cloud keeps $h(f)$ in the memory for future duplicate checks.

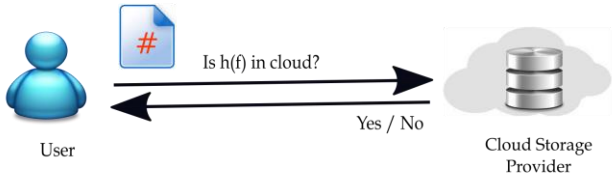
C. General Setting

To backup f, the uploading user u starts with uploading $sh(f)$. The high collision rate of $sh(f)$ turns to be an advantage of brute-force resilience. However, we now turn to face the problem that C needs to blindly determine whether f is deduplicable when receiving only $sh(f)$.

More specifically, the first challenge in the design is to ensure the brute-force resilient deduplication over encrypted



(a) Cross-user cloud envisioned CPS (CCCPS).



(b) Cloud data deduplication (CDD). Fig. 1: The proposed architecture where Fig. 1a is the CCCPS, and Fig. 1b is the cloud data deduplication (CDD).

data with only symmetrically encrypted user-cloud interactions. All existing brute-force resilient solutions rely on certain types of independent servers that assign the proper key to the uploading user. The independent entities are difficult to be removed because the chunk keys need to involve independent randomness; otherwise, the brute-force search will uncover the sensitive content. Here, we take an alternative approach; u generates and stores the chunk key k_f and $E_{k_f}(f)$ in C when f is uploaded for the first time. Otherwise, u retrieves k_f and performs the duplicate check on $E_{k_f}(f)$.

D. Security Model

Storage Model. The cloud storage C performs cross-user chunk-level client-side data deduplication, attempting to decrease storage and network bandwidth costs. The user u possesses an individual key k, which will not be shared with anyone else. The symmetric encryption $E_k(\cdot)$ (e.g., AESCBC), and cryptographic hash function $h(\cdot)$ (e.g., SHA256) are ready for use.

In this work, the user u attempts to store a low min-entropy (i.e., predictable) chunk f in C. The set of all candidate f's is denoted as F. The user u has access to the full-length *chunk hash* $h(f)$ and truncated hash (also called *short hash*) $sh(f)$. Partial bits of $h(f)$ can be considered as a short hash $sh(f)$. Literally, $sh(f)$ has high collision rate and thus the attacker cannot be confident that it is

the specific f that implies $sh(f)$, mitigating the brute-force threat. In essence, u aims to calculate the *chunk key* k_f for encrypting f and to upload $E_{k_f}(f)$.

Threat Model. We consider a legitimate but malicious user $u\hat{}$ (external attacker) and an honest-but-curious cloud C (internal attacker); their common objective is to recover f or k_f owned by the legitimate user u. Here, because the chunk has only low min-entropy, $u\hat{}$ and C may consider *external brute-force* and *internal brute-force* strategies, respectively, to recover f or k_f. In the external (internal) brute-force attack, $u\hat{}$ eavesdrops on (C receives) a deterministic representation of low minentropy content (e.g., $E_k(f)$) and then constructs deterministic representations of all candidate chunks to see which matches the eavesdropped (received) one. Since f is predictable, the degree of randomness in k determines whether the bruteforce can work efficiently. The unauthorized access to the files in the cloud storage due to the system implementation flaw [26] and the design nature of deduplication [8], [27] is a potential security threat. Software updates and Proof-ofOwnership (POW) schemes [27] are developed but are beyond this paper's scope.

As CPS systems involve the collection and exchange of sensitive data between physical devices and cloud infrastructure, **data privacy** and **security** are paramount concerns. For instance, in healthcare, patient data is highly sensitive, and any exposure or unauthorized access could have severe consequences. Similarly, in industrial automation, proprietary data and operational secrets need to be protected from unauthorized access.

Traditional cloud storage systems, even though they offer security features, may not be fully equipped to address the specialized needs of CPS. Common challenges include:

- **Data exposure** during transmission over untrusted networks.
- **Risk of unauthorized access** to stored data, either due to weak encryption or compromised access credentials.
- **Inadequate security during computations**, as cloud providers may be able to inspect data during processing.

1.3 Key Features of LEVER

LEVER's motivation stems from the need to provide end-to-end **encrypted storage** and **secure two-party interactions**, ensuring that data is protected both at rest and in transit. By employing **homomorphic encryption** and **secure multi-party computation**, LEVER allows computations to be performed on encrypted data without exposing the underlying plaintext, thus safeguarding the privacy and integrity of data.

Here, we present our message Lock Encryption with neVerdecrypt homomorphic EncRyption (LEVER). We divide the encrypted data deduplication into three phases; the user derives the chunk key for the chunk to be uploaded in the first phase, transforms the chunk to be uploaded into an encrypted form in the second phase, and then runs the ordinary data deduplication protocol in the third phase.

The design challenge of encrypted data deduplication is that a high min-entropy key can only encrypt the chunk. However, it is still difficult for different users with the same f to calculate the standard high min-entropy key. We have the following criteria for the design of encrypted data deduplication.

- The chunk can only be encrypted by a high min-entropy key, $E_{k_f}(f)$, where k_f is a high min-entropy chunk key for f.
- Since the chunk key k_f needs to be stored in the cloud, a high min-entropy key can only encrypt it

  A critical challenge that arises in cloud storage systems is the inefficiency of storing large volumes of **redundant data**. In many CPS applications, especially those involving sensors and IoT devices, the same data can be recorded multiple times, resulting in **high storage overhead**. For example, a sensor in an industrial plant may generate identical readings multiple times, but without a mechanism to detect and eliminate these duplicates, the storage system will become bloated, leading to unnecessary costs.

  Traditional cloud storage solutions often store duplicate data, which results in:

  - **Wasted storage space** and increased cloud storage costs.
  - **Inefficient data transfer** between devices and the cloud, consuming both bandwidth and processing power.
  - **Longer data retrieval times**, especially when large volumes of duplicate data need to be searched and accessed.

  The **deduplication mechanism** in LEVER provides a solution to these inefficiencies by identifying and eliminating duplicate data at the **block-level**, allowing only unique data to be

stored. This approach reduces storage requirements significantly, improves storage efficiency, and cuts down on cloud service costs. By leveraging deduplication alongside encryption, LEVER ensures that sensitive data remains protected while optimizing storage.

.

Let x be the index calculated and sent by the user who uploads f explicitly. If x is high min-entropy, then the index x can be resistant to the brute-force attack. Here, the index x refers to $\mathcal{E}(k_f \ominus h(f))$ and $E_{E(k_j)}(E_{h(f)}(k_f))$. We hope to have a chunk-dependent key extraction function $g(x,y)$, where x and y denote the materials from the cloud storage and f is the user's chunk, such that $g(x,f)$ extracts the chunk key k_f for f. In the following, we describe how we construct the index x and the chunk-dependent key extraction function $g(x,y)$.

There are two desirable properties of $g(x,f)$.

- $g(x,f)$ extracts the chunk key k_f correctly if the index x is from f.
- $g(x,f)$ outputs a random bit string if the index x is not from f.

In the following, the uploading of a specific chunk can be divided into three cases; the first case refers to that the cloud does not have a copy of the chunk in the consideration and cannot find a match of the short hash in the index table. The second case refers to the cloud not having a copy of the chunk in the consideration but can find a match of the short hash in the index table. The third case refers to that the cloud already has a copy of the chunk in the consideration.

The protocol description of LEVER is shown in Algorithm 1. The algorithm starts with calculating and uploading

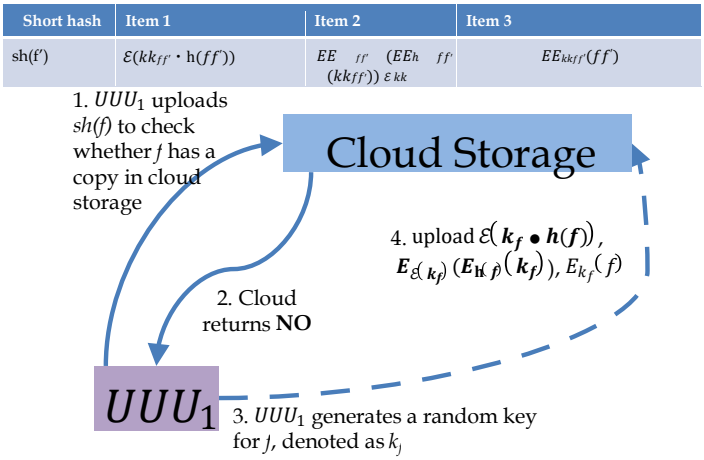
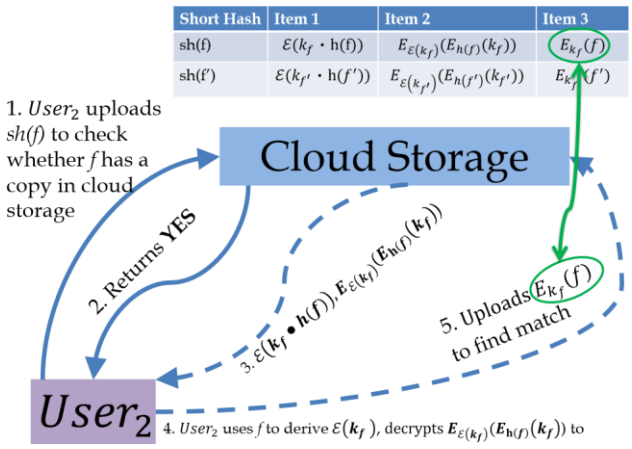


Fig. 2: First uploading of f; $\bullet := \ominus$.



the short hash $sh(f)$ to the cloud storage. Due to the high collision rate, the short hash does not reveal the sensitive content and can be resilient to the brute-force attack. Then, C looks for a match in the index table I (case 1 in Algorithm 1). Line 2 indicates that the cloud cannot find a match of the short hash in the index table. Thus, the user knows that he is the first one who uploads f. The user generates a high minentropy random key k_f (line 3). After that, the user calculates and uploads $E_{k_f}(f)$, and the necessary indexes, $\mathcal{E}(k_f \ominus h(f))$ and $E_{\mathrm{E}(k_f)}(E_{h(f)}(k_f))$ to the cloud (lines 4 and 5). The user keeps k_f in the local storage and erases all of the intermediate materials. Fig. 2 presents the model of case 1 of LEVER.

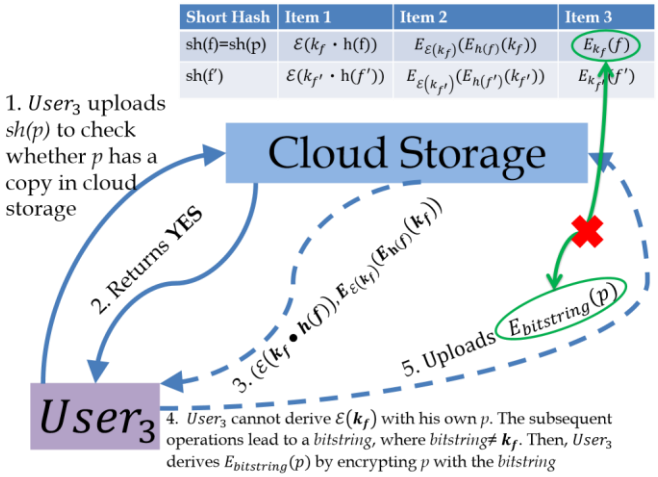
Consider the case of $\mathrm{I}(sh(f)) \neq \emptyset$ (line 6), where C can find at least one match of $sh(f)$ in I. Here, C sends all of the matching records M, where M consists of all the records $[\mathcal{E}(k_{f'} \ominus h(f')), E_{\mathcal{E}(k_{f'})}(E_{h(f')}(k_{f'}))]$ such that $sh(f^0) = sh(f)$, to the user (line 9). Thus, after receiving a positive response, the user knows that someone else probably has uploaded f. The user calculates all of the potential chunk keys for f from the received M. More specifically, for each record $[\mathcal{E}(k_{f'} \ominus h(f')), E_{\mathcal{E}(k_{f'})}(E_{h(f')}(k_{f'}))]$ from M, the user first calculates $\mathrm{E}(h(f))$ (line 9) and then

In this section, we consider the Paillier cryptosystem as an instance of homomorphic encryption. However, we modify the Paillier's scheme in our protocol. In particular, we present the encryption part of Paillier as follow:

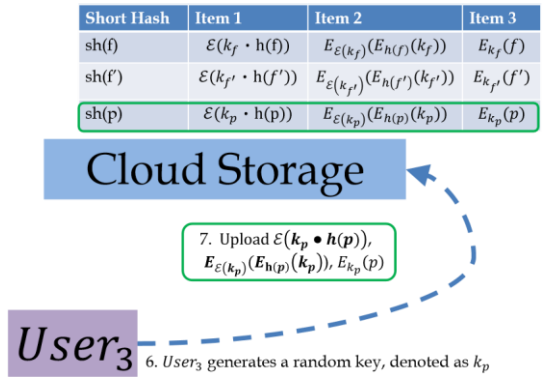
$$\mathrm{E}_r(m) = g^m \cdot r^n \mod n^2, \qquad (1)$$

where g and n are public parameters, r is a random number chosen by the user, and m is the message to be encrypted. Here, we use the notation $\mathrm{E}_r(m)$ to particularly indicate the Paillier encryption of

m is with the specific random number r. When we encrypt m, we are still free to choose random r and perform Eq. 1.



(a) Steps 1-5.



(b) Steps 6-7.

Fig. 4: Uploading of p with $sh(f) = sh(p)$; $\bullet := \ominus$.

Now, consider the case when user 1 uploads the file f and cloud unable to find a match of $sh(f)$ in the index table. Then, user 1 randomly generates a key k_f for f, randomly generates a number r_1, and uploads $\mathcal{E}_{r_1}(k_f \ominus h(f))$ and

$E_{E_{r1}(k_f)}(E_{h(f)}(k_f))$, simultaneously.

Later, user 2 wants to upload the same file f. After exchanging the short hash, cloud returns $\mathcal{E}_{r_1}(k_f \ominus h(f))$ and $E_{E_{r1}(k_f)}(E_{h(f)}(k_f))$ to user 2. Since user 2 has no idea on r_1, so what he can do is to

randomly generate a number r_2 (r_1 6= r_2 with very high probability) and calculate $E_{r_2}(h(f))$. User 2 has the calculated $E_{r_2}(h(f))$ while he received $\mathcal{E}_{r_1}(k_f \ominus h(f))$. Thus, he cannot derive $E_{r_1}(k_f)$ which is required for decrypting $E_{E_{r_1}(k_f)}(E_{h(f)}(k_f)$ which is an issue. To fill this gap, we propose the following method to fix this design problem.

Let us re-consider the above problem as below. User 2 has calculated $E_{r_2}(h(f))$. Besides, user 2 receives $\mathcal{E}_{r_1}(k_f \ominus h(f))$ from the cloud. Notice that user 2 does not know r_1. In terms of Paillier cryptosystem, we know

$$E_{r_1}(m_1) \oplus E_{r_1}(m_2) \quad (2)$$
$$=(g^{m_1} r_1^n) \cdot (g^{m_2} r_2^n) \mod n^2 \quad (3)$$
$$=g_{m_1+m_2}(r_1 r_2)_n \mod n_2 \quad (4)$$
$$=E_{r_1 r_2}(m_1 + m_2) \quad (5)$$

With the above property, we can derive the following relation.

$$\mathcal{E}_{r_2}(h(f)) \oplus \mathcal{E}_{r_1}(k_f \ominus h(f)) = \mathcal{E}_{r_1 r_2}(k_f) \quad (6)$$

In one hand, the user 2 has to check whether $E_{r_1 r_2}(k_f)$ and $E_{r_1}(k_f)$ are both from k_f. However, on the one hand, this is not doable because the authors on [28] proved that probabilistic homomorphic encryption could not allow for equality test, and deterministic homomorphic encryption is not secure. On the other hand, user 2 does not need to check. When we have a closer look, in terms of Paillier cryptosystem, $E_{r_1 r_2}(k_f)$ and $E_{r_1}(k_f)$ can have the following representations.

$$E_{r_1 r_2}(k_f) = g_{k_f}(r_1 r_2)_n \mod n_2 \quad (7)$$

$$\mathcal{E}_{r_1}(k_f) = g^{k_f} r_1^n \mod n^2 \quad (8)$$

During the process, user 2 generates r_2, while it does not know r_1. Also, n is a public parameter in Paillier cryptosystem and known by everyone. Hence, after user 2 derives $E_{r_1 r_2}(k_f) = g^{k_f}(r_1 r_2)^n \mod n^2$, it can calculate the following

$$g^{k_f}(r_1 r_2)^n \cdot (r_2^n)^{-1} \mod n^2, \quad (9)$$

where $(r_2^n)^{-1}$ denotes the inverse element of r_2^n.

In the above procedure, user 2 only needs to compute $(r_2^n)^{-1}$. Obviously, this results in that user 2 obtains $g^{k_f} r_1^n \mod n^2$, which is $E_{r_1}(k_f)$. Moreover, as mentioned in Paillier's original paper [29], $\mathbb{Z}_{n^2}^*$ forms a group, which means that every element in

1.4 Purpose and Scope of Documentation

CPS environments are characterized by **real-time data processing** and **low-latency communication** needs. For example, in autonomous vehicles, sensors need to collect and transmit data to cloud systems almost instantaneously to make real-time decisions that affect vehicle control. In smart cities, sensors deployed across infrastructure must transmit real-time data for urban management and traffic monitoring systems. This creates a need for **efficient, low-latency storage solutions** that can handle large volumes of data generated in real-time.

Traditional cloud storage systems often introduce latency due to the time required for:

- Data upload and download processes.
- Computations on large volumes of encrypted data.
- The transmission of sensitive data through potentially insecure channels.

LEVER was motivated by the need to **minimize latency** while ensuring the **secure transmission** and **real-time availability** of data. By integrating encryption techniques such as **homomorphic encryption**, LEVER enables secure computations directly on encrypted data, avoiding the need for costly decryption steps that can add delays. This approach ensures that CPS applications can meet their **real-time performance requirements** without compromising on data security.

The scalability of cloud-based storage systems is a critical factor in the success of large-scale CPS implementations. CPS applications are inherently **distributed**, with data generated from various devices spread across different locations. As the number of devices increases, the system must be capable of handling increasing amounts of data while maintaining performance and security.

Challenges to scalability include:

- **Handling large-scale data without compromising on performance**: As the number of devices and sensors grows, so does the amount of data that needs to be stored and processed.

- **Managing dynamic and distributed storage environments**: Cloud-based CPS solutions must scale to accommodate growing numbers of devices while ensuring that data is distributed, replicated, and synchronized effectively.

- **Ensuring data security and privacy** at scale, especially when dealing with a large number of devices and potentially untrusted parties.

LEVER was designed to offer a **highly scalable** cloud storage solution that can handle the challenges of large-scale CPS applications. By employing efficient deduplication and secure encryption techniques, LEVER ensures that the system remains both secure and efficient as it scales to accommodate **millions of connected devices**. Additionally, LEVER's architecture supports the **distributed nature** of CPS, allowing for data storage and processing across a wide range of geographically dispersed devices.

The financial burden of cloud storage can be significant, particularly when dealing with high volumes of data. In CPS environments, especially those with long-lived devices like sensors, storing data indefinitely can result in escalating costs. This becomes particularly problematic in IoT-driven applications, where sensor data may be continuously generated, but only a fraction of it may be useful in the long term.

LEVER addresses these concerns by introducing cost-saving mechanisms such as:

- **Data deduplication** to reduce storage overhead, ensuring that only unique data is stored.

- **Encrypted cloud storage** that allows for secure, cost-efficient data management without compromising privacy.

- **Compression techniques** (where applicable) to further reduce storage needs while maintaining data integrity.

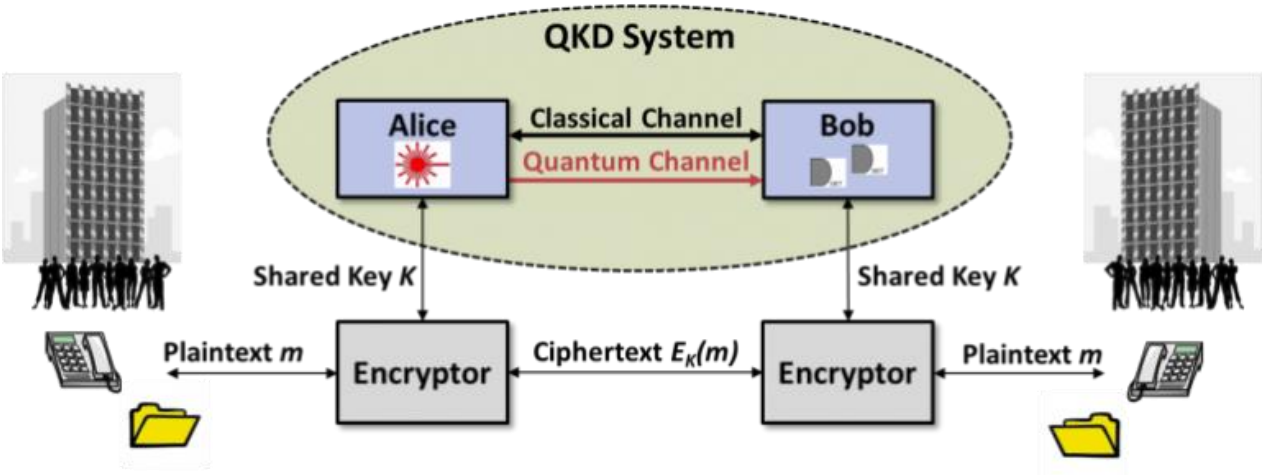
By significantly reducing the amount of redundant data stored, LEVER can help organizations lower **cloud storage costs** while ensuring that data remains secure and accessible.

System Architecture

In this section, first, we describe the evaluation metrics (see Section VI-A). Then, we analytically present the LEVER performance evaluation metrics (see Section VI-B). After that, we describe the scenario in which we conduct our experiments (see Section VI-C). Finally, we present our results in Section VI-D.

In many sectors, such as healthcare, finance, and manufacturing, organizations must comply with stringent data privacy regulations like **GDPR** (General Data Protection Regulation), **HIPAA** (Health Insurance Portability and Accountability Act), and **CCPA** (California Consumer Privacy Act). These regulations often require strict data protection mechanisms, including encryption and secure access control, to ensure that sensitive data is stored and processed in accordance with legal requirements.

LEVER's robust security framework, which includes **end-to-end encryption** and **secure two-party interactions**, ensures that organizations can comply with these regulatory standards while still benefiting from the scalability and flexibility of cloud storage. This compliance aspect is particularly important in industries dealing with highly sensitive data, such as healthcare, where regulatory compliance is critical.



A. Evaluation Metrics

We consider the following two metrics to evaluate the performance of data deduplication techniques. Here, let us assume that the user u uploads a random distinct chunk to the cloud storage C to evaluate the performance. We also assume that uploading requests are uniformly distributed. The metrics are presented below [30].

Communication Cost. The communication cost is reversely proportional to the deduplication percentage in client-side deduplication. However, depending on the protocol design, one may have a subtle difference between different implementations. Such a difference may affect user experience from the user perspective and may aggregate cause performance bottleneck from the cloud point of view.

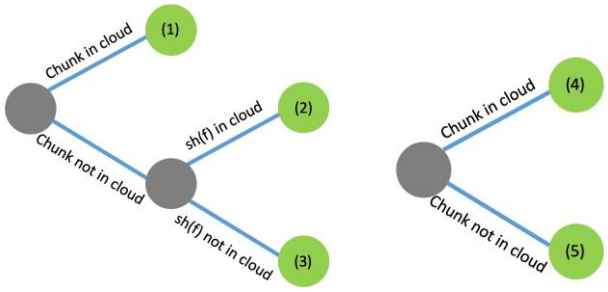
Deduplication Percentage. It is a (normalized) value for evaluating the effectiveness of data deduplication [31] that is defined as $1^{-}{}_{\pi\_\_\_\_o}{}^{1}/_{\pi i}$, where π_o and π_i denote the numbers of bytes input to and output from the deduplicated storage, respectively. The *perfect deduplication* will detect all duplicates and reaches the deduplication ratio of $1 - 1/n = (n - 1)/n$ given n identical files uploaded.

B. Performance Evaluation of LEVER

In here, we explain the metrics and how we apply them on LEVER.

1) Communication Cost $\mathsf{T}_{\mathrm{LEVER}}$*:* Fig. 5a shows the numbers of bits required in the message exchanges of LEVER. From Algorithm 1 and Fig. 5a, we can understand that in the case of $\mathrm{I}(sh(f)) = \emptyset$, u has to spend $(`_k \cdot `_{sh} + `_{sh} + `_k + `_f)$ bits to upload $E_{k_f}(f)$, and the required information in I. On the opposite, $\mathrm{I}(sh(f))$ 6= $\emptyset$, the exact number of bits are $`_f$. Furthermore, in order to ease the calculation, we also assume that $Pr[sh(f)$ 6= $\emptyset | f /\in S] = 1$. Hence, $\mathsf{T}_{\mathrm{LEVER}}$ formula is defined as

$$\begin{aligned}\mathsf{T}_{\mathrm{LEVER}} &= \mathrm{P}(`_f + (`_k \cdot `_{sh} + `_k)|\mathrm{M}|) \\ &+ (1 - \mathrm{P})((`_k \cdot `_{sh} + `_k)|\mathrm{M}|) + `_k \cdot `_{sh} + `_{sh} + `_k + `_f).\end{aligned} \tag{10}$$

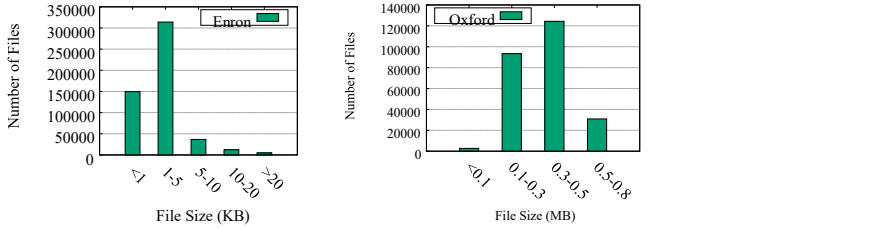


(a) Communication Cost (b) Deduplication percentage

Fig. 5: LEVER metrics where (1): $\grave{}_f + (\grave{}_k \cdot \grave{}_{sh} + \grave{}_k)|M|$, (2): $(\grave{}_k \cdot \grave{}_{sh} + \grave{}_k)|M|) + \grave{}_k \cdot \grave{}_{sh} + \grave{}_{sh} + \grave{}_k + \grave{}_f)$, (3): $\grave{}_k \cdot \grave{}_{sh} + \grave{}_{sh} + \grave{}_k + \grave{}_f)$, (4): l_f, and (5): $\grave{}_k \cdot \grave{}_{sh} + \grave{}_{sh} + \grave{}_k + \grave{}_f$.

2) Deduplication Percentage D_{LEVER}*:* Fig. 5b presents the occupied spaces in different cases of the uploading behaviors in the LEVER approach. As a consequence, D_{LEVER} can be approximated as

$$D_{LEVER} = 1 - \frac{1}{\ell_f/((\mathcal{P}(\ell_f)) + (1-\mathcal{P})(\ell_k \cdot \ell_{sh} + \ell_{sh} + \ell_k + \ell_f))}. \quad (11)$$



(a) Enron (b) Oxford

Fig. 6: The file size distribution used in our experiment. Fig. 6a: Enron email dataset [12] and Fig. 6b: Oxford buildings dataset [13].

C. Scenario Description

This section presents LEVER's exhaustive evaluation compared to the original data deduplication (O-DD) and without data deduplication (W-DD) strategies. We test our experiment on two datasets, namely Enron email dataset [12] and Oxford building landmark dataset [13] that we illustrate their traces in Fig. 6. The Enron dataset contains emails, and the Oxford dataset includes multimedia content. We select these datasets because we believe that ordinary users have the demand to backup the email and multimedia content to the CCPS. We conduct our test on Apple Mac 64 bit OS Catalina version 10.12 on Intel Core i5 2.9 GHz and 8GB of RAM. We run the code on Python 3.7.6 platform. We use the SHA-256 hash function on the OpenSSL library. Moreover,

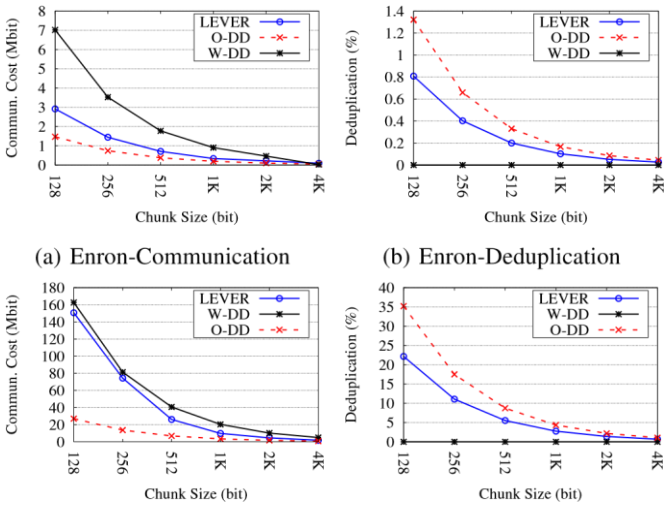
we use the symmetric cipher AES and Pallier encryption scheme [32]. We follow the details of file settings from our previous work [33]. The python implementation of LEVER is available in [14].

D. Numerical Simulations

In this paper, we present our numerical results to validate LEVER based on the deduplication percentages and communication cost metrics, as described in Section VI-A.

1) Impact of chunk size (l_f): In the first experiment, we test the communication cost and deduplication percentage based on varying chunk sizes. We compare LEVER with original data deduplication (O-DD) and without deduplication (W-DD) methods.

To do this, in this experiment, we consider Enron and Oxford datasets which have different data volumes, fix the short hash to 8 (bit) (i.e., $l_{sh} = 8$) and probability persistence of each chunk to 0.3 (i.e., P = 0.3) and show the results in Fig. 7. From this figure, we can understand *three* main conclusions. First, when the chunk size increases, the communication cost and the deduplication percentage for both of the LEVER and O-DD protocols in both datasets decreases. It confirms that the LEVER method could manage the deduplication rate the same as the original deduplication but with a very little higher communication cost. Also, O-DD has lower communication costs than LEVER because at least the user and server in LEVER, for security reasons, have heavier communications. Thanks for the LEVER, which provides secure data encryption for each chunk; hence, it helps the results be more reliable than O-DD. Second, the deduplication percentage of both LEVER and O-DD protocols is similar, especially for the Enron



(a) Enron-Communication (b) Enron-Deduplication

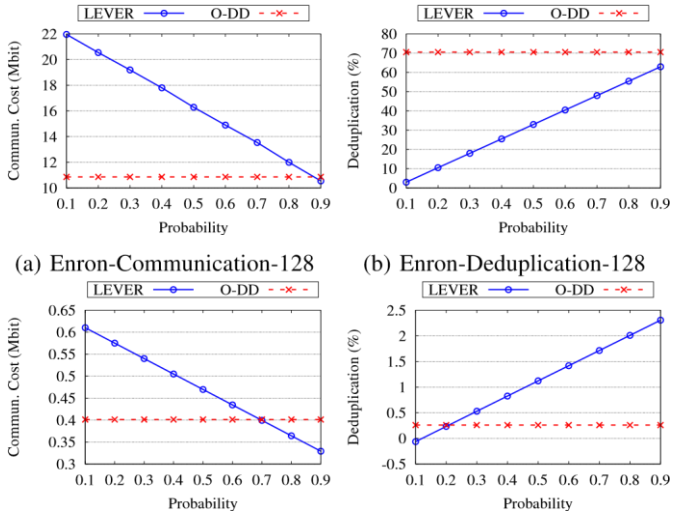
(c) Oxford-Communication (d) Oxford-Deduplication

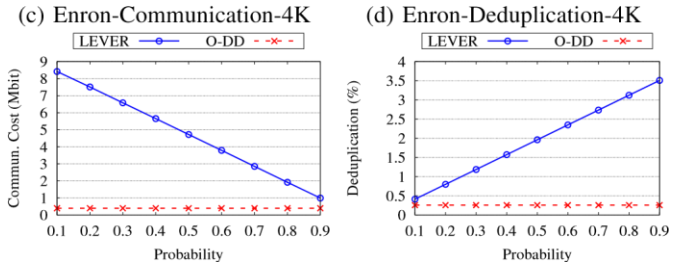
Fig. 7: Results for the impact of chunk size (l_f) where Figs. 7a and 7c are the communication cost and Figs. 7b and 7d are deduplication percentage among LEVER, W-DD, and O-DD for l_{sh} = 8 and P = 0.3 for Enron and Oxford datasets.

dataset. Third, the communication and data deduplication costs for the Oxford dataset are higher than the Enron dataset. It is because the volume of the Oxford dataset is much higher than the Enron dataset. In the end, we should highlight that WDD does not consider deduplication; hence, its deduplication percentage is zero (see the black continues curves in Figs. 7b and 7d).

*2) Impact of persistence probability of a chunk (*P*):* In the second experiment, we compare our LEVER communication and deduplication costs with the O-DD protocol by considering varying the probability and a fixed chunk sized. In this experiment, we set the $l_f = \{128, 4K\}$ (bit), $l_{sh} = 8$, and the P = {0.1,0.9}. We present the results in Fig. 8. The results indicate some interesting concepts. First, when we increase the probability percentage of a chunk available in cloud storage, we can see that the LEVER's communication cost decreases when it compares against O-DD. Second, by increasing the probability, the deduplication percentage is growing. It is because of that cloud storage will involve more in data deduplication to save bandwidth and storage. Third, by increasing the P and increasing 32X of the chunk size, the communication/deduplication cost of LEVER converges faster to the O-DD protocol. Note that LEVER depends on the various P, while O-DD is independent of P. At the same time, it confirms the significant performance of our method.

3) Impact of short hash length (l_{sh}): In the next experiment, we show the comparison between LEVER and O-DD protocol based on varying the size of a short hash. To do this, we test the experiments for 8-bit and 16-bit size hash length (i.e., $l_{sh} = \{8,16\}$). We show the results in Fig. 9. Focusing on Figs. 9a and 9c, from the LEVER value we understand when we fix the probability to 0.3, and the value of l_{sh} increasing 2x, the communication cost for both chunk sizes increases. The main reason is that the size of a short hash affects the communication cost because more data volume





(e) Oxford-Communication-4K (f) Oxford-Deduplication-4K

Fig. 8: Results for the impact of probability (P) in Enron/Oxford datasets where (i) Enron dataset: Figs. 8a and 8c are the communication cost and Figs. 8b and 8d are deduplication percentage and (ii) Oxford dataset: Figs. 8e and 8f are the communication and deduplication cost among LEVER and O-DD with 128 and 4K chunk sizes in bits for $l_{sh} = 8$ and $P \in [0.1, 0.9]$.

will be transferred to the CDD in a CPS. In contrast, it is fixed for O-DD (it is independent of short hash type). It confirms that LEVER could manage the communication by increasing the short hash length, especially for 128-bit chunk sizes. Focusing on Figs. 9b and 9d, it is evident that LEVER deduplication value decreases when we increase the value of the P. The decrement rates for the deduplication rate reduces when we increase the l_{sh}. This decrease is because increasing the short hash length leads to fewer collisions between short hashes. Hence, cloud storage could decide more accurately to do deduplication or normal uploading. Therefore, it leads to decreasing the deduplication rate.

Summary of the result achievement: As we can conclude from the tested experiments on both datasets, LEVER is a feasible and reliable cloud data deduplication solution between two users located in various CPSs. Additionally, LEVER can protect data when brute-force imposes to the cloud data for both low-entropy (in the Enron dataset) and high-entropy (in Oxford dataset) contents and enhance the bandwidth protection.

LEVER is a cutting-edge cloud storage solution designed to address the unique challenges faced by **Cyber-Physical Systems (CPS)**, particularly in the realms of data security, storage efficiency, and real-time processing. Below are the key features that make LEVER a powerful and innovative system:

1. Encrypted Two-Party Interactions

One of LEVER's standout features is its ability to **securely manage interactions between two parties** (typically the data owner and cloud service provider) through **encrypted communication protocols**. LEVER ensures that:

- **Data Privacy**: Data remains encrypted during transmission and while being processed by the cloud provider, meaning no sensitive information is exposed during communication.

- **Confidentiality**: Even if the cloud provider's infrastructure is compromised, the data remains unreadable and protected due to strong encryption mechanisms.

- **Integrity**: LEVER guarantees that the data transmitted between two parties is not altered or tampered with, ensuring data integrity throughout the interaction.

This feature is essential in ensuring that sensitive data, such as IoT sensor outputs, medical records, or proprietary industrial data, remains private and intact.

2.Secure Data Deduplication

Data deduplication is a critical function for optimizing storage in environments where large volumes of potentially redundant data are generated. LEVER introduces a secure method of **deduplicating data in the cloud** while maintaining encryption, which addresses the common issues of traditional deduplication, such as:

- **Redundant Data Elimination**: LEVER identifies duplicate copies of data, ensuring that only unique data is stored in the cloud. This significantly reduces the storage footprint, leading to reduced costs.

- **Encryption-Aware Deduplication**: LEVER can deduplicate encrypted data without needing to decrypt it, a feature that preserves privacy while optimizing storage efficiency.

This is a challenge for traditional deduplication systems, which often require data to be decrypted for comparison.

- **Storage Optimization**: By removing duplicates, LEVER enhances storage efficiency, especially important in CPS where large volumes of data are generated by numerous sensors and devices.

3. End-to-End Encryption

LEVER guarantees **end-to-end encryption** for both data at rest and data in transit:

- **Data at Rest**: All stored data is encrypted using strong encryption algorithms, ensuring that even if unauthorized parties gain access to the cloud storage, they cannot read or manipulate the data.

- **Data in Transit**: Data transmitted between users and cloud services is encrypted during the entire transfer process, protecting it from potential eavesdropping or tampering.

- **Encryption Algorithms**: LEVER supports both symmetric and asymmetric encryption techniques, providing flexibility in how data is secured. Additionally, **homomorphic encryption** allows for computations to be performed on encrypted data without decrypting it, ensuring privacy even during processing.

This encryption strategy ensures that data is protected at all stages of its lifecycle, addressing privacy concerns and regulatory requirements (e.g., GDPR, HIPAA).

4. Homomorphic Encryption for Privacy-Preserving Computation

LEVER utilizes **homomorphic encryption**, a powerful cryptographic technique that enables computations to be performed on encrypted data without the need to decrypt it. This feature is particularly important for:

- **Privacy-Preserving Data Analysis**: Sensitive data can be analyzed without ever being exposed in its plaintext form. This is useful in scenarios where organizations need to process personal or confidential data but cannot afford to expose it to external parties, including the cloud provider.

- **Secure Multi-Party Computation (SMC)**: Homomorphic encryption enables secure computation in distributed environments. Multiple parties can jointly compute results without revealing their individual data, ensuring that privacy is maintained throughout the interaction.

- **Compliance with Regulations**: By ensuring that data is never decrypted during processing, LEVER helps meet regulatory requirements that demand high standards of data privacy and protection.

5. Real-Time Data Processing

CPS environments often require **real-time data processing** due to the dynamic and time-sensitive nature of the data. LEVER is optimized to handle:

- **Low Latency**: LEVER minimizes delays in processing and transmitting data, ensuring that data can be securely stored and accessed with minimal latency. This is crucial in applications like autonomous vehicles, industrial automation, and healthcare, where real-time decision-making is necessary.

- **Efficient Cloud Interaction**: LEVER efficiently handles the secure transmission and processing of large volumes of data, even from geographically distributed devices. This ensures that CPS can function without delay, providing a seamless user experience.

By supporting **real-time cloud storage** and computation, LEVER meets the stringent performance demands of real-time CPS applications.

6. Scalable and Distributed Architecture

LEVER is designed to be **scalable**, allowing it to handle the massive amounts of data generated by CPS applications, which often involve thousands or even millions of connected devices. Key aspects of its scalability include:

- **Distributed Storage**: LEVER's architecture is distributed across multiple nodes or cloud regions, enabling it to store large volumes of data securely across multiple servers or data centers. This distributed approach ensures high availability, fault tolerance, and load balancing.

- **Elastic Scalability**: LEVER can scale up or down based on the requirements of the CPS environment. Whether the system needs to handle a sudden surge in data or maintain consistent performance as it grows, LEVER adapts to meet demand.

- **Efficient Resource Utilization**: Through deduplication and intelligent data storage management, LEVER optimizes the use of cloud resources, ensuring that storage costs remain manageable as the system scales.

This scalability ensures that LEVER can support both small-scale CPS applications and large, complex systems involving millions of devices.

7. Interoperability with Existing CPS Ecosystems

LEVER is built to integrate seamlessly with existing **CPS infrastructures**, including a wide variety of IoT devices, sensors, actuators, and other components. The system supports:

- **Cross-Platform Integration**: LEVER can interface with different cloud platforms and service providers, allowing organizations to use existing infrastructure while enhancing it with LEVER's secure and efficient storage capabilities.

- **IoT Device Compatibility**: LEVER is optimized to handle the specific challenges of storing and processing data from IoT devices, which often operate in highly distributed and dynamic environments.

- **APIs and SDKs**: LEVER provides **developer-friendly APIs** and software development kits (SDKs) that allow easy integration with external systems, facilitating data synchronization and secure storage across a wide range of CPS applications.

This interoperability makes LEVER an attractive solution for organizations looking to enhance their existing CPS infrastructure without overhauling it entirely.

8. Cost-Efficiency

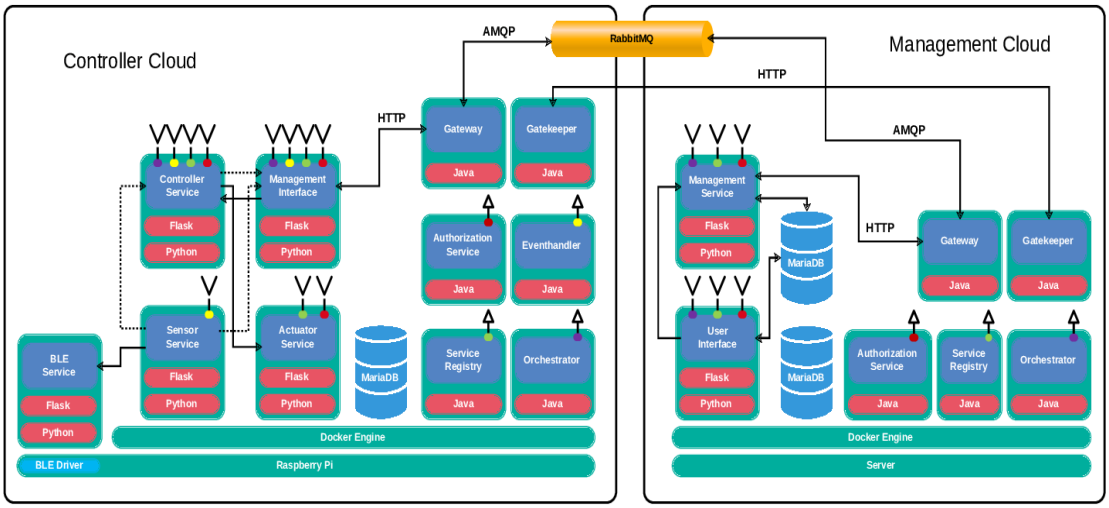
LEVER's design emphasizes cost-effectiveness by:

- **Optimized Storage**: Through deduplication, LEVER minimizes the amount of storage required, reducing cloud service costs significantly.

- **Efficient Data Transfer**: By reducing redundant data and only transferring unique data, LEVER minimizes bandwidth usage and associated costs.

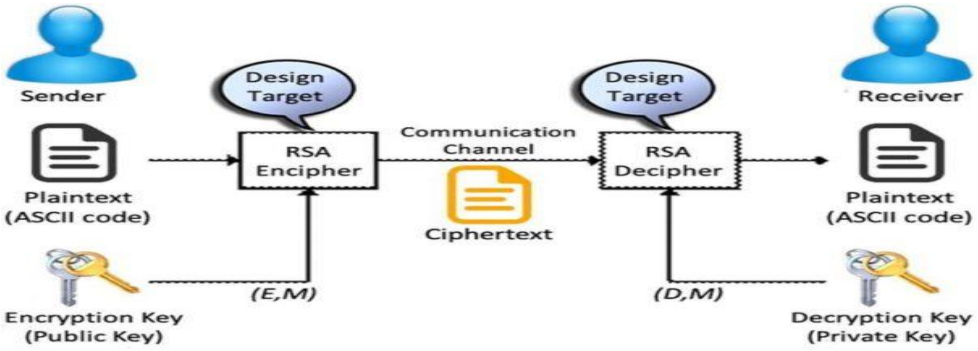
- **Elastic Pricing Models**: LEVER supports flexible, cloud-based pricing models that align with the user's actual storage needs, ensuring that the system can be used cost-effectively regardless of scale.

The cost-efficient architecture ensures that LEVER delivers value to organizations, particularly those in resource-constrained environments or industries where operational costs need to be minimized.


Controller Cloud
AMQP
RabbitMQ
Management Cloud
HTTP
HTTP
Gateway
Gatekeeper
Java
Java
Controller Service
Flask
Python
Management Interface
Flask
Python
Authorization Service
Java
Eventhandler
Java
AMQP
Management Service
Flask
Python
MariaDB
HTTP
Gateway
Java
Gatekeeper
Java
Sensor Service
Flask
Python
Actuator Service
Flask
Python
MariaDB
Service Registry
Java
Orchestrator
Java
User Interface
Flask
Python
MariaDB
Authorization Service
Java
Service Registry
Java
Orchestrator
Java
BLE Service
Flask
Python
Docker Engine
BLE Driver
Raspberry Pi
Docker Engine
Server

Cryptographic Models and Protocols

Cryptographic models and protocols are the foundational elements that ensure secure communication and data protection in various systems, including computer networks, cloud computing, and distributed systems. These models and protocols provide the theoretical framework and practical means to achieve confidentiality, integrity, authentication, and non-repudiation.



1. Cryptographic Models:

Cryptographic models are theoretical frameworks that describe how cryptographic systems are constructed and evaluated. They help in understanding the security properties and the assumptions underlying cryptographic algorithms. Here are some common cryptographic models:

- **Symmetric-Key Model**: In symmetric-key cryptography, the same key is used for both encryption and decryption. The security of the model depends on the secrecy of the key and the ability of both parties to securely exchange or share this key. Common examples are AES (Advanced Encryption Standard) and DES (Data Encryption Standard).

- **Asymmetric-Key Model (Public-Key Cryptography)**: This model uses two keys—one public and one private. The public key is used for encryption, while the private key is used for decryption. The security of this model relies on the difficulty of certain mathematical problems, such as factoring large numbers (RSA) or solving discrete logarithms (ECC). Examples include RSA and Elliptic Curve Cryptography (ECC).

- **Hybrid Model**: Combines the strengths of both symmetric and asymmetric encryption. For example, in a secure communication session, asymmetric encryption can be used to exchange a symmetric key, and then symmetric encryption can be used to encrypt the actual data.

- **Zero-Knowledge Proofs (ZKPs)**: A cryptographic protocol that allows one party to prove to another that they know a value without revealing the value itself. This concept is particularly important in privacy-preserving protocols.

2. Cryptographic Protocols

Cryptographic protocols define the procedures for securely transmitting data or performing cryptographic operations. They ensure that security goals like confidentiality, integrity, authentication, and non-repudiation are met. Some key cryptographic protocols include:

- **Secure Sockets Layer (SSL) / Transport Layer Security (TLS)**: SSL/TLS are cryptographic protocols designed to provide secure communication over a computer network. They use a combination of asymmetric and symmetric encryption to ensure confidentiality and integrity of the data in transit.

- **Public Key Infrastructure (PKI)**: A framework that uses asymmetric cryptography to enable secure exchange of information over insecure networks. PKI uses digital certificates and a certificate authority (CA) to authenticate identities and facilitate secure communication.

- **Diffie-Hellman Key Exchange**: A cryptographic protocol that allows two parties to securely exchange cryptographic keys over a public channel. The exchanged keys can later be used for symmetric encryption.

- **Digital Signature Algorithm (DSA)**: A protocol for creating and verifying digital signatures. It ensures data integrity and authenticity by signing a message with a private key, which can later be verified by others using the corresponding public key.

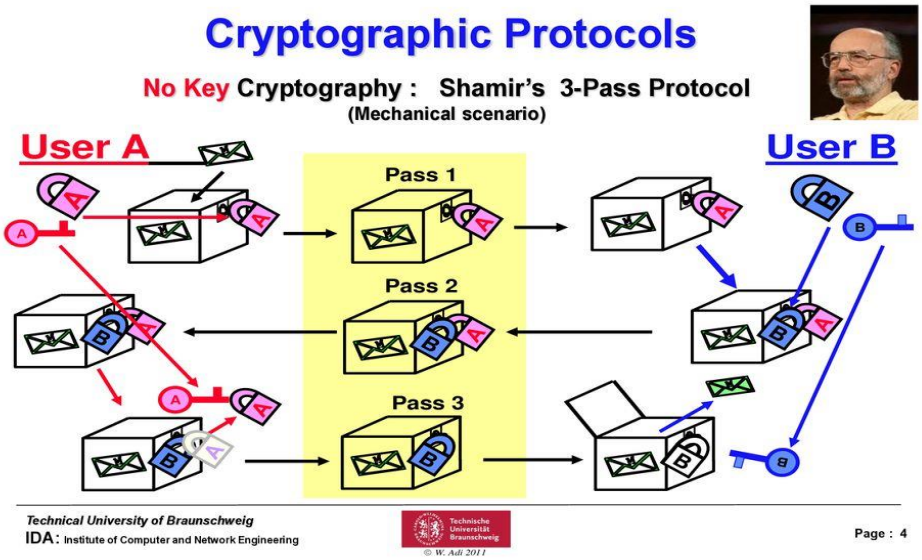
- **IPsec (Internet Protocol Security)**: A suite of protocols used to secure Internet Protocol (IP) communications by authenticating and encrypting each IP packet in a communication session.

    - **Authentication Protocols**: **Kerberos**: A network authentication protocol that uses symmetric encryption and a trusted third party (the Key Distribution Center) to authenticate users and services in a network.

    - **OAuth**: A protocol for token-based authentication, commonly used in web applications to allow secure delegated access to resources without sharing passwords.

- **Secure Multi-Party Computation (SMPC)**: A protocol that enables parties to compute a joint function over their inputs without revealing their individual inputs. It's used in privacy-preserving applications.

- **Shamir's Secret Sharing**: A cryptographic protocol used to distribute a secret among a group of participants. The secret can only be reconstructed if a certain number of participants cooperate.



3. Cryptographic Goals and Properties

Cryptographic protocols aim to achieve several important goals:

- **Confidentiality**: Ensuring that only authorized parties can access the information. This is typically achieved using encryption algorithms.
- **Integrity**: Ensuring that the data has not been altered during transmission. This is typically achieved using cryptographic hash functions and digital signatures.
- **Authentication**: Ensuring that the identity of parties in a communication is verified. This is often accomplished using digital certificates, password-based protocols, or biometrics.
- **Non-Repudiation**: Ensuring that once a transaction or message is sent, the sender cannot deny having sent it. This can be achieved with digital signatures and logs.
- **Forward Secrecy**: A property of cryptographic protocols that ensures that session keys are not compromised even if long-term keys are later exposed.

4. Applications of Cryptographic Models and Protocols

- **Secure Communication**: Encryption protocols like SSL/TLS and IPsec are used in web browsing, email, and virtual private networks (VPNs) to ensure secure transmission of data.
- **Blockchain and Cryptocurrencies**: Cryptographic protocols, including hash functions and digital signatures, are used to secure transactions and maintain the integrity of distributed ledgers.
- **Digital Payments**: Payment systems like credit cards, mobile wallets, and online banking rely on cryptographic protocols to protect transactions from fraud and theft.

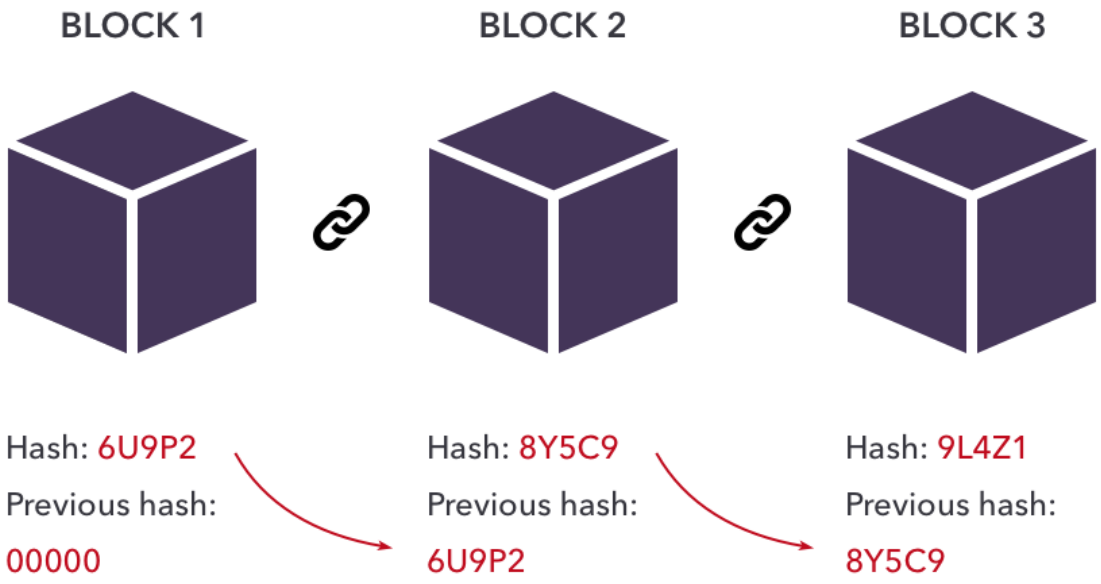
- **Identity and Access Management**: Authentication protocols like OAuth, OpenID, and Kerberos help secure user identities and manage access control in networks and applications.

5. Challenges and Considerations

- **Key Management**: The generation, distribution, and storage of cryptographic keys is a critical challenge in cryptography. Poor key management practices can undermine the security of a system.

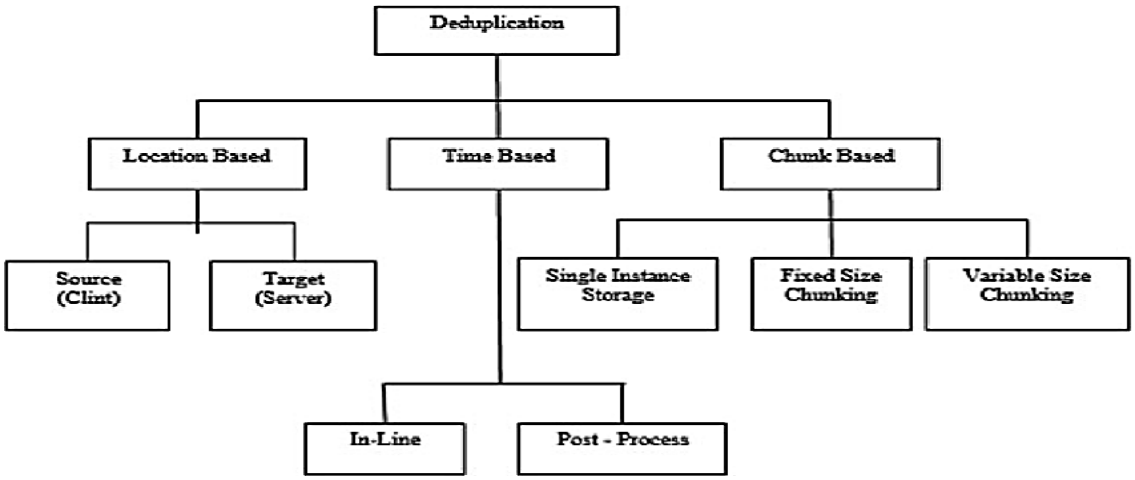
- **Quantum Computing Threats**: The advent of quantum computing could break many existing cryptographic algorithms (e.g., RSA, ECC). This has led to the development of quantum-resistant algorithms, such as lattice-based cryptography.

- **Side-Channel Attacks**: Cryptographic protocols can be vulnerable to attacks that exploit physical or timing information, such as power consumption or electromagnetic radiation. Secure implementations must consider these risks.



Deduplication Mechanisms

Deduplication mechanisms are techniques used in data storage and transmission to eliminate redundant copies of data. By identifying and removing duplicate data, these mechanisms can optimize storage space, reduce network bandwidth usage, and improve the efficiency of backup and recovery systems. Deduplication is commonly used in file storage, backup systems, and cloud storage to ensure data is stored or transmitted efficiently.



1. Types of Deduplication:

There are several types of deduplication mechanisms, based on when and how they operate. The most common classifications are:

- **File-Level Deduplication**:
    - **Description**: This type of deduplication compares entire files to identify duplicates. If two or more files are identical, only one copy is stored, and references are made to the stored file in place of the duplicates.
    - **Usage**: File-level deduplication is most useful when many identical files exist, such as in scenarios where multiple copies of the same document or image are stored.

- **Block-Level Deduplication**:
    - **Description**: Instead of comparing entire files, block-level deduplication divides data into smaller segments, typically of fixed sizes (e.g., 4KB or 8KB), and removes duplicate blocks of data. Even if the files themselves are different, if they share the same block, the system only stores one copy of that block.
    - **Usage**: Block-level deduplication is more efficient than file-level deduplication for large data sets and scenarios where only parts of files are common (e.g., backup systems or virtual machine images).
- **Inline Deduplication**:
    - **Description**: Inline deduplication happens in real-time, as data is written to storage. Each chunk of data is compared to existing chunks before it's written, ensuring that only unique data is stored. If a duplicate is found, it's discarded or replaced with a reference to the original.
    - **Usage**: Inline deduplication is beneficial when real-time data optimization is required, and it's commonly used in modern storage systems, particularly in cloud services.
- **Post-Process Deduplication**:
    - **Description**: In this approach, data is written to storage first, and deduplication happens later during a scheduled process. After the initial storage of data, the system scans the data and removes duplicates.
    - **Usage**: Post-process deduplication is typically used in backup systems or archives, where data is stored first, and the removal of duplicates can be done periodically, not in real time.

2. Deduplication Techniques

- **Hashing**: Deduplication is commonly achieved by using hashing algorithms (e.g., SHA-1, SHA-256). Each block of data or file is hashed, and the hash value is used as a fingerprint. When new data is added, its hash is compared to existing hashes in the storage system. If the hash matches an existing one, the data is considered a duplicate and is not stored again.

- **Fingerprints**: Fingerprinting is a variation of hashing where a "fingerprint" or "signature" of a file or data block is generated. When a new file or block is stored, the system compares its fingerprint to the stored ones. If a match is found, the system avoids storing the duplicate data.

- **Chunking**: In block-level deduplication, chunking is the process of dividing data into smaller chunks (blocks) before comparison. The chunking algorithm defines how the data is split, and if multiple chunks are identical, they are deduplicated. Chunking can be **fixed-size** (uniform block sizes) or **variable-size** (using a technique like Rabin fingerprinting to divide data at boundaries based on content).

3. Benefits of Deduplication

- **Storage Savings**: The most obvious benefit of deduplication is the significant reduction in storage requirements. By eliminating duplicate data, organizations can store more information in the same amount of space.

- **Reduced Network Bandwidth Usage**: When used in backup or data transfer scenarios, deduplication can reduce the amount of data transmitted over the network, which is particularly valuable in cloud storage environments where bandwidth costs may be high.

- **Faster Backups and Restores**: Deduplication can improve backup and restore performance by only transferring and storing unique data, making it faster to back up data to storage and recover it when needed.

- **Improved Storage Efficiency**: In environments where large volumes of data are stored, especially for backup and archival purposes, deduplication can vastly improve storage efficiency by only keeping unique data blocks.

- **Cost Savings**: By reducing the amount of storage needed, organizations can cut hardware costs and operational expenses for storing large amounts of data.

4. Challenges of Deduplication

- **Processing Overhead**: Deduplication, especially in inline processes, introduces processing overhead as data needs to be analyzed and compared before being written. This can add latency to operations, particularly in high-performance environments.

- **Complexity in Managing Metadata**: Deduplication requires the management of extensive metadata (e.g., hash values or fingerprints) to track which data blocks are unique and which are duplicates. This can increase the complexity of storage systems, especially for large datasets.

- **Data Fragmentation**: In some cases, block-level deduplication can lead to data fragmentation, where data is split into smaller, non-contiguous blocks. This can affect read/write performance if not properly managed.

- **Challenges in Deduplicating Encrypted Data**: Encrypted data, due to its randomized nature, cannot be deduplicated effectively because the encryption process changes the data every time it's encrypted. To solve this, deduplication is often applied before encryption in secure storage solutions.

- **Loss of Redundancy**: In some scenarios, removing duplicates can lead to a loss of redundancy. For example, if a file is deduplicated and only a single copy is stored, losing access to that copy could mean a complete loss of data, unless replication or backup systems are in place.

5. Use Cases for Deduplication

- **Backup and Disaster Recovery**: Deduplication is widely used in backup systems to reduce the amount of storage required for backup copies. For example, in an enterprise environment, backup systems often store incremental or differential backups, where only the changes from the previous backup are stored. Deduplication helps reduce the storage requirements of these backups.

- **Cloud Storage**: Cloud storage services use deduplication to minimize the amount of data uploaded and stored, reducing storage costs for both providers and customers. It also helps improve upload and download speeds by avoiding the transfer of duplicate data.

- **Email Systems**: Deduplication is applied in email systems, where identical attachments or messages are common. By storing a single copy of these emails and referencing it where needed, storage usage can be optimized.

- **Virtual Machine (VM) Environments**: In VM environments, where virtual disk images may be identical across multiple virtual machines, deduplication helps reduce the storage footprint by eliminating duplicates in the VM storage.

- **File Sharing and Collaboration**: In file-sharing environments, especially with cloud services or corporate file servers, deduplication ensures that only one copy of a file is stored, even if multiple users upload the same file.

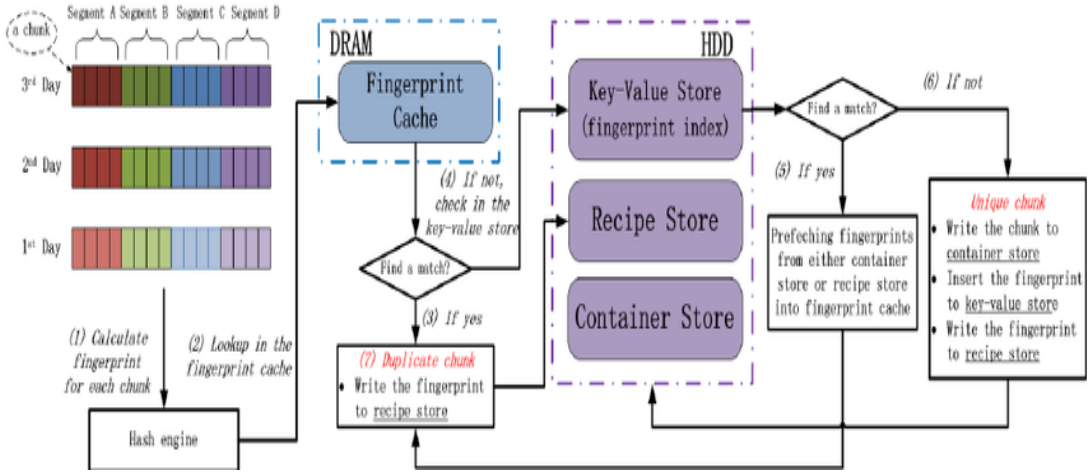
6. Popular Deduplication Tools and Technologies

- **NetApp ONTAP**: NetApp's storage solution uses inline and post-process deduplication for both primary and backup storage systems.

- **Dell EMC Data Domain**: A leading backup appliance that integrates deduplication to optimize storage and backup processes.

- **Veritas NetBackup**: Offers deduplication capabilities for backup systems, enabling efficient storage of backup data.

- **Veeam Backup & Replication**: Provides both source-side and target-side deduplication for efficient backup storage in virtual environments.



Instead of comparing entire files, block-level deduplication divides data into smaller segments, typically of fixed sizes (e.g., 4KB or 8KB), and removes duplicate blocks of data. Even if the files themselves are different, if they share the same block, the system only stores one copy of that block.

In VM environments, where virtual disk images may be identical across multiple virtual machines, deduplication helps reduce the storage footprint by eliminating duplicates in the VM storage. In file-sharing environments, especially with cloud services or corporate file servers, deduplication ensures that only one copy of a file is stored, even if multiple users upload the same file

Privacy and Security Features

Privacy and security features are essential components in protecting sensitive data and ensuring the confidentiality, integrity, and availability of information. These features are implemented across various platforms, systems, and applications to safeguard against unauthorized access, data breaches, and cyber threats. Below, we'll discuss the key privacy and security features, how they work, and their importance in modern computing systems.

1. Encryption

Encryption is one of the most fundamental security features. It transforms data into an unreadable format that can only be reverted back to its original form using a decryption key. This ensures confidentiality and prevents unauthorized users from accessing sensitive information.

- **Symmetric Encryption**: The same key is used for both encryption and decryption (e.g., AES – Advanced Encryption Standard).
- **Asymmetric Encryption**: Uses a pair of keys — a public key for encryption and a private key for decryption (e.g., RSA, Elliptic Curve Cryptography).

Encryption is used in various contexts, including:

- **File Encryption**: Encrypting files on disk to prevent unauthorized access.
- **Transport Encryption**: Protocols like **TLS (Transport Layer Security)** and **SSL (Secure Sockets Layer)** ensure encrypted communication over networks (e.g., HTTPS for websites).
- **End-to-End Encryption**: Common in messaging apps (e.g., WhatsApp, Signal) to ensure that only the sender and receiver can read the message content.

2. Authentication and Authorization

Authentication is the process of verifying the identity of a user, device, or system, while authorization determines the level of access granted.

- **Multi-Factor Authentication (MFA)**: Combines multiple factors (something you know, something you have, and something you are) to authenticate a user. This greatly enhances security.
    - Example: Using a password (something you know) combined with a one-time code from an app (something you have) or a fingerprint (something you are).
- **Single Sign-On (SSO)**: Allows users to authenticate once and gain access to multiple systems without needing to log in repeatedly.
- **Biometric Authentication**: Uses unique biological features, such as fingerprints, facial recognition, or retina scans, to authenticate users.
- **Role-Based Access Control (RBAC)**: Limits system access to authorized users based on their role within an organization, ensuring that users only have access to the information necessary for their role.

3. Data Integrity

Data integrity ensures that data remains accurate and unaltered during storage, transmission, or processing. Techniques used to ensure data integrity include:

- **Hash Functions**: A hash is a one-way function that converts data into a fixed-size string of characters. Hashes are used to verify the integrity of data (e.g., MD5, SHA-256). When transmitting data, a hash value of the data is generated, and upon receipt, the hash is checked to see if the data has been tampered with.

- **Digital Signatures**: Used for ensuring both data integrity and authenticity. A digital signature is a cryptographic value calculated from the data and a private key. The recipient can verify it using the public key to confirm that the data hasn't been altered and comes from the expected sender.

4. Privacy-Preserving Techniques

Privacy features focus on protecting the user's personal data and ensuring that individuals have control over what information they share.

- **Anonymization**: The process of removing personally identifiable information (PII) from datasets so individuals cannot be identified. This is commonly used in healthcare, research, and public datasets.

- **Pseudonymization**: Replacing personal identifiers with pseudonyms, which can be reversed if needed with the proper key. This is often used in environments where data needs to be processed but still require a level of protection.

- **Data Minimization**: The principle of collecting only the data necessary for a specific purpose, thereby reducing the amount of personal information at risk.

- **Privacy by Design**: A design framework that emphasizes the integration of privacy features from the start of a product's development, rather than as an afterthought.

5. Firewall and Intrusion Detection Systems (IDS)

Firewalls and IDS are security mechanisms that protect networks and systems from unauthorized access and cyberattacks.

- **Firewall**: Acts as a barrier between trusted internal networks and untrusted external networks (e.g., the internet). It monitors incoming and outgoing traffic, enforcing security rules based on IP addresses, ports, or protocols.

- **Intrusion Detection Systems (IDS)**: Monitors network or system activities for malicious activities or policy violations. IDS can be **signature-based** (detects known threats) or **anomaly-based** (detects deviations from normal behavior).

- **Intrusion Prevention Systems (IPS)**: A step beyond IDS, IPS actively blocks or prevents detected threats in real-time.

6. Secure Software Development Practices

Ensuring that applications are designed, developed, and deployed with security in mind is essential to protect against vulnerabilities.

- **Code Auditing and Static Analysis**: Automated tools and manual processes are used to identify vulnerabilities and weaknesses in the application code, such as SQL injection, cross-site scripting (XSS), or buffer overflow vulnerabilities.

- **Secure Coding Guidelines**: Adhering to established secure coding practices (e.g., OWASP Top 10) helps developers avoid common vulnerabilities and secure their code against potential threats.

- **Patch Management**: Ensuring that software is regularly updated to patch known security vulnerabilities is critical. Unpatched software is a common entry point for attackers.

7. Data Loss Prevention (DLP)

DLP tools monitor and prevent the unauthorized sharing or leakage of sensitive information.

- **Content Inspection**: DLP solutions inspect data as it moves through networks or storage systems and can block or alert on attempts to share sensitive information (e.g., credit card numbers, social security numbers).

- **Endpoint Protection**: Ensures that devices like laptops, smartphones, and workstations adhere to security policies and prevent the transfer of unauthorized data.

8. Backup and Recovery

Robust backup and recovery solutions ensure that data can be restored after a disaster or breach. Features typically include:

- **Encryption of Backup Data**: Ensuring backup data is encrypted both at rest and in transit to prevent unauthorized access.
- **Automated Backups**: Scheduled or continuous backups of critical systems and data to ensure availability and minimize the impact of data loss.
- **Disaster Recovery Plans**: Procedures and tools in place to recover data and systems after a cyberattack, natural disaster, or hardware failure.

9. Access Control and Auditing

Access control features ensure that only authorized users can access specific resources or data. Auditing features help track user activities for compliance and security purposes.

- **Audit Logs**: Logs of user actions, system events, and security incidents are maintained to provide a detailed record of activities. These logs can help detect malicious behavior and are essential for compliance (e.g., GDPR, HIPAA).
- **Access Control Lists (ACLs)**: Lists that define permissions and access rights for users and groups to specific resources (files, networks, etc.).
- **Privileged Access Management (PAM)**: Controls access to critical systems and data by limiting the use of privileged accounts (admin, root) and enforcing least privilege access.

10. Compliance with Regulations and Standards

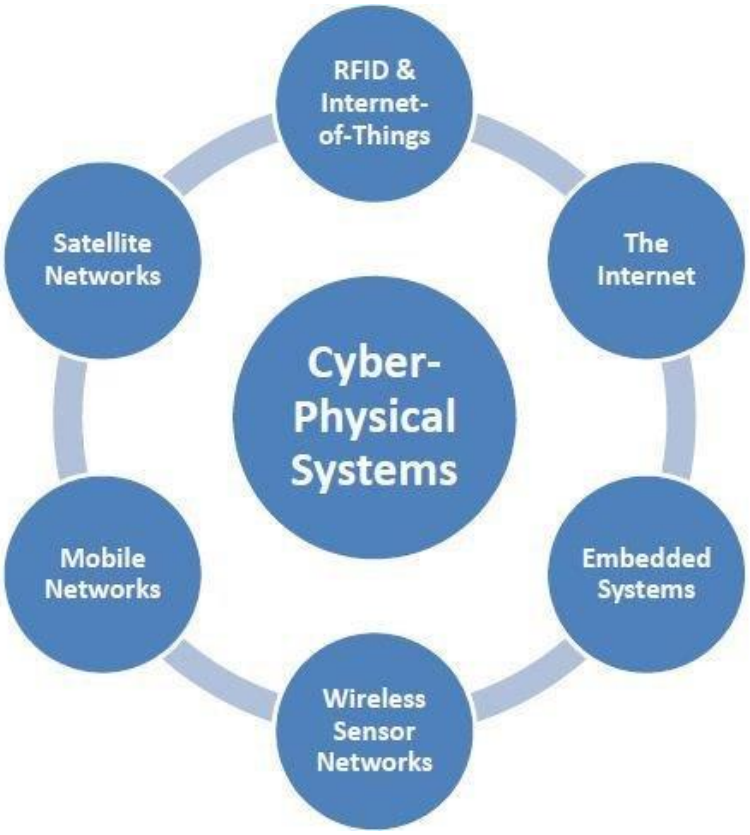
Many privacy and security features are built to comply with industry regulations and standards, which govern the handling of sensitive data. These regulations ensure that organizations protect data properly and respect user privacy rights.

- **General Data Protection Regulation (GDPR)**: A European Union regulation that requires organizations to protect the privacy and personal data of EU citizens and gives individuals more control over their data.
- **Health Insurance Portability and Accountability Act (HIPAA)**: A U.S. regulation that mandates the protection and confidential handling of medical data.
- **Payment Card Industry Data Security Standard (PCI DSS)**: A global standard for securing credit card transactions and protecting cardholder information.
- **ISO/IEC 27001**: A standard for information security management systems (ISMS) that provides guidelines for managing and securing information assets.

Cyber-Physical Systems and LEVER Integration

Cyber-Physical Systems (CPS) are systems that involve the integration of computational elements with physical processes. These systems combine computer-based algorithms with physical processes and have a significant impact on various industries, including manufacturing, healthcare, transportation, and energy. CPS is at the heart of the Internet of Things (IoT), where devices and machines communicate, monitor, and control physical processes through a networked digital infrastructure.

LEVER integration, on the other hand, refers to leveraging various technological and system-level integrations to enhance the functionality of CPS. It focuses on integrating different components of a system, such as hardware, software, networks, and data analytics, to make CPS more efficient, adaptable, and resilient.



1. Cyber-Physical Systems (CPS) Overview

a) Definition and Components

A Cyber-Physical System is a network of physical entities (sensors, actuators, machines) connected through a digital computational infrastructure (computers, cloud systems, communication networks). The system interacts with the physical world through sensors that monitor the system and actuators that take actions in the physical environment.

Key components of CPS:

- **Physical Elements**: Sensors, actuators, and devices that monitor and interact with the physical world. For example, temperature sensors, pressure sensors, GPS devices, robotic arms, etc.

- **Computational Infrastructure**: Embedded software, cloud computing resources, and data processing units that manage, analyze, and respond to the data received from the physical world. This includes processors, algorithms, and systems for decision-making.

- **Communication Networks**: Networks (wired and wireless) that connect the computational infrastructure and physical devices, allowing real-time communication. Examples include wireless sensor networks, the internet, and low-power wide-area networks (LPWAN).

- **Data Analytics and Decision Making**: The computational component that processes incoming data, performs analytics, and makes decisions that affect the system's behavior. This often involves machine learning, control algorithms, and AI.

b) Applications of CPS

Cyber-Physical Systems are utilized across various industries:

- **Smart Manufacturing**: CPS enables smart factories where machines and robots communicate to optimize production lines and improve efficiency. Example: Predictive maintenance systems that monitor machine health and prevent breakdowns.

- **Healthcare Systems**: Remote monitoring of patients through wearable devices (e.g., smartwatches) that communicate with healthcare professionals in real time.

- **Smart Transportation**: Autonomous vehicles and connected infrastructure that improve safety, reduce congestion, and optimize traffic flow.

- **Energy Systems**: Smart grids and energy management systems that integrate renewable energy sources, track energy usage, and improve efficiency.

2. LEVER Integration in CPS

LEVER stands for **Leveraging and Integrating** the various technologies and processes involved in the system to ensure that the CPS functions efficiently and adaptively in real-time. LEVER integration is essential to make sure that all components of a Cyber-Physical System work seamlessly together. It focuses on creating systems that can handle complexities, uncertainties, and adaptability by ensuring that all elements—hardware, software, networks, and analytics—are harmoniously integrated.

a) Key Aspects of LEVER Integration

LEVER integration involves the following principles:

1. **Hardware-Software Integration**: Ensuring that embedded hardware and the software running on it are tightly coupled to ensure that sensors, actuators, and processors can interact efficiently.

2. **Communication Integration**: Ensuring seamless communication between all components of a CPS, including sensors, devices, and cloud systems. LEVER integration takes into

account different communication protocols, data exchange formats, and latency requirements.

3. **Data Integration**: Combining data from different sources (sensors, cloud platforms, historical data) and ensuring data consistency and compatibility. Data analytics tools, including AI and machine learning algorithms, are crucial for making sense of the data.

4. **Control Integration**: Ensuring that control algorithms and decision-making processes are integrated into the physical systems in a way that allows for real-time adjustments based on sensor feedback.

5. **Security Integration**: Ensuring that the entire CPS ecosystem is secure by implementing cybersecurity measures at both the hardware and software levels. This involves encryption, secure communication channels, and secure firmware updates.

6. **Resilience and Adaptability**: Leveraging redundancy, fault-tolerance, and adaptive systems to ensure that CPS can continue to function effectively even in the case of component failures, unexpected changes, or evolving conditions.

b) Technologies Involved in LEVER Integration

The integration process in CPS involves several technologies working together, including:

- **Internet of Things (IoT)**: IoT devices like smart sensors and connected actuators that continuously collect data from the physical world and transmit it to computational units for processing.

- **Cloud Computing**: The use of cloud infrastructure for storing large volumes of data generated by CPS, running complex data analytics, and providing decision support for managing physical systems.

- **Edge Computing**: In cases where low-latency decisions are necessary, edge computing ensures that data processing happens close to the source (on the devices themselves or local

computing nodes), reducing dependency on cloud infrastructure and enabling faster responses.

- **Artificial Intelligence (AI) and Machine Learning (ML)**: Machine learning algorithms are used to analyze data, detect patterns, and make predictions that inform decision-making within CPS. AI also allows for autonomous operation and optimization.

- **Blockchain**: For security and data integrity, blockchain technology can be used to create immutable logs of transactions and interactions within CPS, ensuring transparency and preventing tampering with data.

- **5G Networks**: High-speed, low-latency communication networks like 5G are crucial for ensuring real-time data exchange and synchronization between physical and computational components.

3. Challenges in Cyber-Physical Systems and LEVER Integration

While integrating and leveraging technologies to build CPS, several challenges need to be addressed:

- **Complexity**: Integrating diverse components (hardware, software, sensors, networks) creates complexity in managing and coordinating the system. Keeping track of system states, coordinating real-time actions, and managing data flows are non-trivial.

- **Interoperability**: CPS often involve multiple vendors, each using different standards and technologies. Ensuring that components from different vendors can work together seamlessly is a significant challenge.

- **Data Management**: With massive amounts of real-time data generated by CPS, efficiently storing, processing, and analyzing data becomes a huge challenge. Ensuring that data is consistent, accurate, and actionable in real-time is vital.

- **Security and Privacy**: The interconnected nature of CPS makes them susceptible to cyberattacks. Ensuring data confidentiality, protecting communication channels, and safeguarding against potential cyber threats are major concerns.

- **Scalability**: As the size and complexity of CPS grow, ensuring that systems can scale and handle increased loads (data, devices, users) is crucial.

4. Benefits of LEVER Integration in CPS

By successfully integrating the various components of CPS, several benefits can be realized:

- **Increased Efficiency**: Optimizing processes, reducing waste, and improving the reliability of systems.

- **Better Decision-Making**: Leveraging real-time data for informed decision-making and automation.

- **Real-time Monitoring and Control**: Enabling systems to react instantaneously to changing conditions.

- **Cost Savings**: By improving the efficiency and longevity of physical assets (e.g., through predictive maintenance), CPS can help organizations save on operational costs.

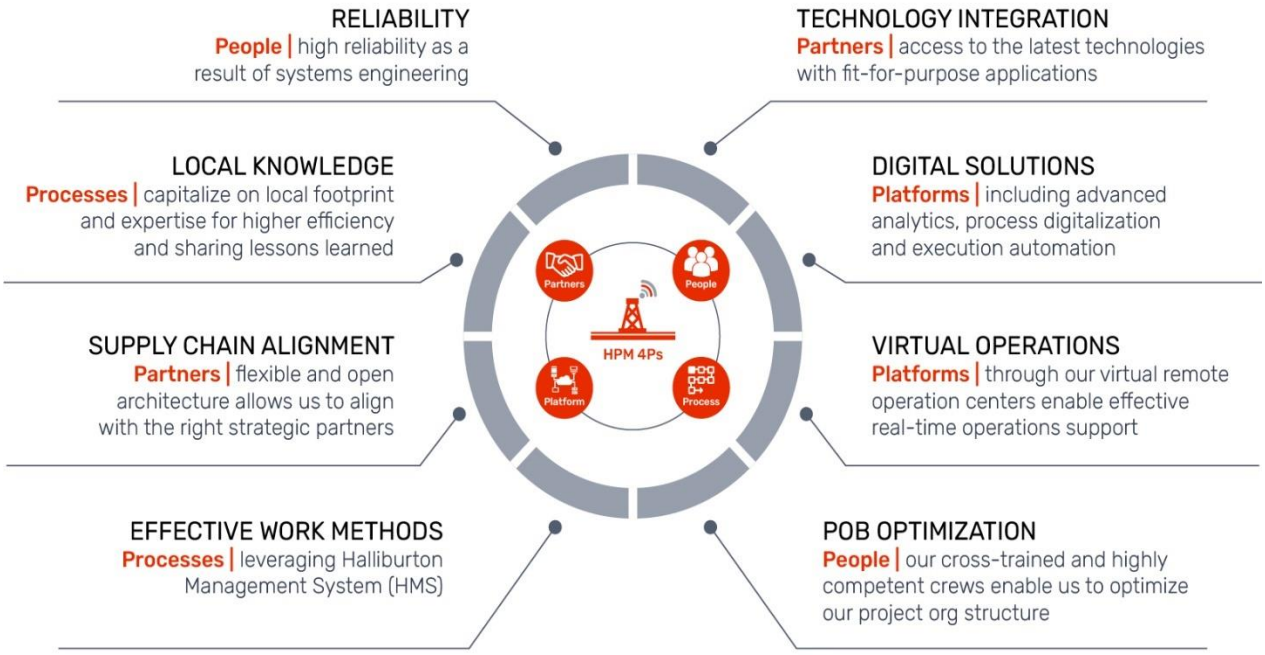
- **Improved Safety**: Through real-time monitoring and autonomous control, CPS can detect potential safety risks and mitigate them before they cause harm.

5. Examples of CPS and LEVER Integration

1. **Smart Grids**: In a smart grid, CPS integrates sensors, communication networks, and computational algorithms to monitor and control the distribution of electricity. LEVER integration ensures that power distribution is efficient, reduces wastage, and incorporates renewable energy sources.

2. **Autonomous Vehicles**: Autonomous vehicles use CPS to integrate sensors (e.g., cameras, lidar), real-time data processing, AI algorithms for decision-making, and communication with infrastructure (e.g., traffic lights, road signs). LEVER integration ensures the safe operation of autonomous systems.

3. **Industrial Automation (Industry 4.0)**: In manufacturing, CPS enables factory machines, robots, and assembly lines to communicate and work together in real time. LEVER integration combines machine learning algorithms, robotics, and predictive maintenance to enhance production efficiency and reduce downtime.



LEVER stands for **Leveraging and Integrating** the various technologies and processes involved in the system to ensure that the CPS functions efficiently and adaptively in real-time. LEVER integration is essential to make sure that all components of a Cyber-Physical System work seamlessly together. It focuses on creating systems that can handle complexities, uncertainties, and adaptability by ensuring that all elements—hardware, software, networks, and analytics—are harmoniously integrated.

Performance Analysis

Performance analysis is the process of assessing and evaluating the efficiency, effectiveness, and overall performance of a system, application, or process. It involves examining various metrics, such as speed, resource utilization, responsiveness, throughput, and reliability, to determine how well a system is functioning and identify potential areas for improvement.

Performance analysis is used in a wide range of fields, including computing systems, software development, networking, and business processes, to optimize system resources, improve user experience, and meet performance requirements.

- **Throughput**: The rate at which a system processes tasks or data. For example, the number of transactions processed per second or the amount of data transmitted over a network.

- **Latency**: The time it takes for a system to respond to a request or trigger an action. In networking, this refers to the time taken for a data packet to travel from sender to receiver.

- **Response Time**: The total time from when a user initiates a request until the system returns a response. It is particularly important in web applications and real-time systems.

- **Resource Utilization**: The amount of system resources (CPU, memory, storage, etc.) used during operation. High resource utilization could indicate inefficiency or the need for optimization.

- **Scalability**: The ability of a system to handle increased loads by scaling up (adding more resources to a single node) or scaling out (adding more nodes or distributed resources).

- **Reliability and Availability**: Measures how consistently the system performs its intended function, and how often it is available for use. This includes fault tolerance and the ability to recover from failures.

- **Power Consumption**: In embedded systems or CPS, the power efficiency of the system can be critical. Monitoring energy consumption is essential for battery-powered devices or environmentally conscious systems.

- **Error Rate**: The frequency of errors or failures in the system during operation. A high error rate can signal bugs, hardware issues, or configuration problems.

- **Synthetic Benchmarks**: Predefined tests designed to measure specific aspects of system performance (e.g., CPU performance, memory bandwidth).

- **Real-World Benchmarks**: Tests based on real application workloads to simulate how the system performs in actual use.

- **CPU Profiling**: Measures how much time the CPU spends on various tasks or functions in the program.

- **Memory Profiling**: Tracks memory usage to detect memory leaks or excessive consumption that could affect system performance.

- **I/O Profiling**: Monitors input/output operations (e.g., disk read/write, network requests) to identify slow-performing areas.

Implementation of LEVER in Cyber-Physical Systems (CPS)

The **LEVER** concept focuses on **Leveraging and Integrating** various technologies, systems, and processes to create efficient, scalable, and resilient Cyber-Physical Systems (CPS). CPS integrates computational elements with physical processes, and the implementation of LEVER aims to ensure seamless collaboration between all components of such systems. LEVER emphasizes the importance of integrating hardware, software, communication protocols, data analytics, and security to enable CPS to function optimally in real-time and under varying conditions.

In this context, **LEVER integration** refers to the process of bringing together different technologies and systems (such as sensors, actuators, computing platforms, communication networks, and control algorithms) into a unified, efficient, and scalable CPS. Below, we explore the key steps involved in the **implementation of LEVER** in Cyber-Physical Systems.

The first step in implementing LEVER is defining the **specific requirements** and **objectives** of the CPS. This includes understanding the **functional requirements**, the **physical processes** involved, and the **system constraints** (e.g., resource limitations, real-time requirements, scalability, etc.).

- **Identify Key Stakeholders**: Gather input from stakeholders such as engineers, end-users, researchers, and business leaders.

- **Define Functional Requirements**: What tasks should the CPS accomplish? For example, in a smart grid system, the functional requirement could be to optimize energy distribution.

- **Set Performance Criteria**: Establish metrics for performance such as latency, throughput, reliability, scalability, and energy efficiency.

- **Constraints Analysis**: Identify constraints related to hardware limitations, power consumption, real-time data processing, network bandwidth, and security considerations.

- The next step is designing the **system architecture** that specifies how the different components will work together. The architecture must address both the **physical** and **computational** elements, as well as how they will communicate.

Key considerations for system architecture:

- **Embedded Systems**: For CPS, embedded devices such as sensors and actuators need to be integrated into the system. These devices should have low power consumption and sufficient computational capabilities to handle basic tasks.

- **Communication Networks**: The architecture must include communication networks (e.g., Wi-Fi, LoRa, 5G) to transmit data between components. The communication layer should ensure low latency, reliability, and real-time performance.

- **Cloud/Edge Computing**: The system should leverage cloud or edge computing to store, process, and analyze large volumes of data generated by the physical processes.

- **Control Algorithms**: Control algorithms that allow the CPS to interact with the physical world and make autonomous decisions need to be integrated into the architecture.

- **Data Integration**: The system should enable seamless integration of data from various sources (e.g., sensors, IoT devices, historical data, external systems) to enable informed decision-making.**LEVER integration** emphasizes a smooth integration of the **hardware** and **software** components of the system. This is where the physical sensors, actuators, and computational platforms come together.

Hardware Integration:

- **Sensors and Actuators**: These are the physical devices that monitor and interact with the environment. Sensors measure environmental variables like temperature, humidity, pressure, and more. Actuators perform actions based on the data received from sensors, such as adjusting machinery or controlling lights.

- **Embedded Systems**: Embedded platforms such as microcontrollers (e.g., Arduino, Raspberry Pi) are used to control sensors and actuators.

- **Connectivity Modules**: These modules allow embedded systems to communicate with each other and with higher-level systems (e.g., via Wi-Fi, Zigbee, or cellular networks).

Software Integration:

- **Firmware Development**: Write embedded software (firmware) for microcontrollers to interface with sensors and actuators.

- **Middleware**: Middleware is essential for handling communication between different devices and ensuring seamless data exchange between sensors, actuators, and computational components. Middleware solutions can include protocols such as MQTT, CoAP, or HTTP.

- **Control Software**: Software for real-time control and decision-making algorithms. These systems handle the interpretation of sensor data, execution of control commands, and adjustment of actuators.

- Communication protocols define how data is exchanged between different parts of the CPS. Since the components in CPS often use different technologies, **LEVER integration** must ensure that the system can efficiently communicate and share data across networks.

- **Protocol Selection**: Choose communication protocols based on system needs. For real-time communication, protocols like **MQTT** or **DDS (Data Distribution Service)** may be used. For low-power, long-range communications, **LoRa** or **NB-IoT** could be ideal.

- **Data Flow Management**: Design systems for how data flows between sensors, embedded devices, cloud services, and user interfaces. Data should be processed and analyzed in real time to provide insights and trigger actions without delay.

- **Latency Considerations**: Implement low-latency communication methods to ensure timely data transmission and real-time decision-making. Latency could be critical in applications like autonomous vehicles or industrial control systems.

- **Edge and Cloud Integration**: Use **edge computing** (processing closer to the sensors and actuators) to reduce communication overhead and ensure faster decision-making in time-sensitive applications. Use **cloud computing** for more extensive data analysis, storage, and remote monitoring.

- **Data processing** and **analytics** play a crucial role in CPS, enabling it to make intelligent decisions based on real-time data from the physical world. **LEVER integration** involves integrating the computational resources needed to process and analyze data.

- **Real-Time Data Processing**: Implement algorithms that can process incoming data from sensors in real time. Use **stream processing** and **edge computing** for quick responses.

- **Data Analytics**: Leverage machine learning and AI techniques to analyze large datasets, identify patterns, and make predictions. For example, predictive maintenance in industrial CPS can predict equipment failure based on historical data and sensor readings.

- **Decision Support**: Integrate decision-making systems that interpret processed data and execute control actions, whether autonomously or under human supervision. This may involve **control theory** (e.g., PID control), decision trees, or reinforcement learning.

Challenges in Cyber-Physical Systems (CPS) and LEVER

1. **Complexity in System Design and Integration**

    - **Multi-disciplinary Requirements**: CPS involves integrating physical processes with computational systems, combining knowledge from several disciplines such as mechanical engineering, computer science, control theory, and networking. Designing such a system that works seamlessly requires collaboration among experts in diverse fields, which can complicate the development process.

    - **Heterogeneity of Components**: CPS often use a wide variety of sensors, actuators, and devices, which may have different communication protocols, power requirements, and computational capacities. Ensuring interoperability among these devices and components is a significant challenge.

    - **Scalability**: As CPS are often deployed in large-scale environments (e.g., smart cities, industrial IoT systems), ensuring that the system can scale efficiently while maintaining performance is a major challenge. As the system grows, managing the increased load on networks, computational resources, and storage becomes more complex.

2. **Real-Time Processing and Latency Issues**

    - **Real-Time Decision-Making**: Many CPS applications, such as autonomous vehicles or industrial automation systems, require real-time data processing to make critical decisions. The complexity of the underlying algorithms and the real-time constraints make it challenging to ensure that decisions are made within the required time limits.

    - **Network Latency**: In distributed CPS, especially those that rely on cloud or edge computing, communication latency can be a critical issue. Network delays can

impact real-time processing and lead to suboptimal decisions, making low-latency communication protocols essential.

- **Data Synchronization**: In multi-component systems, maintaining accurate synchronization of data streams from sensors and actuators is critical to avoid inconsistencies or conflicts, especially in dynamic environments where conditions change rapidly.

3. **Security and Privacy Concerns**

   - **Cybersecurity Threats**: As CPS become more connected, they become increasingly vulnerable to cyber-attacks such as denial of service (DoS), man-in-the-middle (MITM) attacks, and data tampering. Ensuring the security of communication networks and embedded devices is essential to prevent system breaches that could have serious consequences, particularly in critical infrastructure like healthcare or transportation.

   - **Privacy Protection**: Many CPS applications (e.g., smart homes, healthcare) collect sensitive data from users. Ensuring the privacy of this data and complying with privacy regulations (e.g., GDPR, HIPAA) is an ongoing challenge, especially when data is shared across different entities or platforms.

   - **Access Control and Authentication**: With many components in a CPS network, ensuring proper authentication and authorization for access to devices and data is challenging. Implementing effective access control mechanisms that prevent unauthorized access and manipulation is a crucial security concern.

4. **Power Consumption and Energy Efficiency**

   - **Energy Constraints**: Many CPS devices, particularly in IoT environments, are energy-constrained and operate on batteries or limited power sources. Optimizing energy consumption while maintaining performance is a key challenge, especially

when dealing with real-time or mission-critical applications where performance cannot be sacrificed.

- **Power-Hungry Devices**: As more computational resources (e.g., AI, machine learning models) are pushed to the edge for real-time processing, the energy demands on devices increase. Managing energy consumption while ensuring the system remains operational is essential for long-term sustainability.

- **Energy Harvesting and Sustainability**: In some applications, like environmental monitoring or remote sensors, power harvesting techniques (such as solar or kinetic energy) can be utilized. However, these solutions are often limited by environmental factors and may not provide consistent energy supply.

5. **Data Management and Integration**

   - **Big Data Challenges**: CPS generate vast amounts of data from sensors, devices, and external sources. Storing, processing, and analyzing this data in real time to derive actionable insights presents a significant challenge. The complexity of managing big data while maintaining performance, security, and privacy is a key barrier to effective CPS deployment.

   - **Data Quality and Consistency**: In CPS, the accuracy and reliability of data from sensors are crucial for decision-making. Sensor inaccuracies, noise, or failures can affect the overall system performance. Ensuring high-quality, consistent data, especially in dynamic or changing environments, is essential.

   - **Data Fusion**: Integrating data from various heterogeneous sources (e.g., different sensors, external data streams) to create a unified view of the system is difficult. Effective data fusion techniques are needed to combine different types of data (e.g., spatial, temporal, sensory) and provide meaningful insights for real-time decision-making.

Future Work and Solutions to Overcome Challenges

1. **Improved Integration Frameworks and Standards**

    - The development of universal integration frameworks and standards for CPS is critical to solving the interoperability challenges. Standardizing communication protocols, data formats, and APIs can significantly simplify the integration of heterogeneous devices and components across CPS.

    - Collaborative efforts between industry stakeholders, regulatory bodies, and academia will play a crucial role in defining these standards and ensuring that they are widely adopted.

2. **Advancements in Edge and Fog Computing**

    - To address the issues of real-time processing and latency, **edge computing** and **fog computing** can be further developed. By processing data closer to the source (at the edge of the network), these approaches can reduce latency and bandwidth requirements. Future work should focus on optimizing edge and fog computing solutions for more efficient data processing, resource allocation, and real-time decision-making in CPS.

    - **Edge AI** (deploying machine learning algorithms at the edge) will also become more prominent in the future, enabling real-time decision-making in distributed environments without relying on cloud computing.

3. **Enhanced Security and Privacy Techniques**

    - As CPS become more interconnected, improving cybersecurity and privacy protection will be paramount. Future work should focus on developing advanced encryption algorithms, secure communication protocols, and intrusion detection systems tailored to the specific needs of CPS.

- Blockchain and **distributed ledger technologies** could play a role in enhancing data integrity, privacy, and trust in CPS. By ensuring transparent, immutable, and secure transactions across devices, these technologies can help mitigate risks associated with data breaches and unauthorized access.

4. **Energy-Efficient Technologies**

   - Research into **low-power hardware** and energy-efficient algorithms will be essential to address the energy challenges faced by CPS. Future work should focus on optimizing power management techniques, including **dynamic voltage scaling**, **energy harvesting**, and **sleep modes** to extend the battery life of IoT devices and reduce the overall energy consumption of CPS.

   - Leveraging **AI-based energy optimization** algorithms can help monitor and optimize energy usage in real-time based on system performance and environmental conditions.

5. **Artificial Intelligence and Machine Learning for Automation**

   - AI and **machine learning** (ML) will play an increasing role in optimizing CPS performance, especially in real-time decision-making. ML algorithms can be used to predict system failures, optimize energy consumption, enhance fault tolerance, and improve predictive maintenance strategies.

   - **Autonomous CPS**, such as self-driving cars or smart factories, will benefit greatly from AI and ML technologies. Future work should focus on making AI more interpretable and explainable to ensure trust and reliability in critical applications.

Conclusion

Cyber-Physical Systems (CPS) are rapidly transforming industries by integrating the physical world with computational processes to enable real-time decision-making, automation, and enhanced operational efficiency. The implementation of **LEVER integration**, which involves leveraging and seamlessly integrating various technologies, systems, and components, plays a pivotal role in enabling the full potential of CPS. However, the complexity, scalability, real-time processing requirements, security, and privacy challenges faced during the integration of hardware, software, communication protocols, and control systems must be addressed effectively to ensure the success of CPS deployments.

In this context, we have explored the key components of CPS and LEVER integration, including the design and architecture of these systems, the need for advanced data analytics, the importance of real-time decision-making, and the role of security and privacy measures. Each of these aspects requires careful planning and execution to create systems that are not only functional and efficient but also secure, resilient, and adaptable to changing environments.

Despite the substantial progress made, several challenges persist in the development and deployment of CPS, including difficulties in system integration, data management, energy efficiency, and maintaining high levels of security and privacy. The heterogeneous nature of CPS components and the demand for real-time, low-latency decision-making systems add to the complexity. Moreover, as these systems scale up, they need to efficiently manage large volumes of data and ensure robust communication among a vast network of devices.

Looking forward, there is a clear need for advancements in several areas to overcome these challenges. These include the development of standardized protocols for interoperability, the implementation of edge and fog computing to reduce latency.

References

1. **Lee, E. A.** (2008). Cyber-Physical Systems: Design Challenges. *Proceedings of the 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, 363–369. https://doi.org/10.1109/ISORC.2008.15

2. **Pantelopoulos, A., & Bourbakis, N. G.** (2010). A Survey on Wearable Sensor-Based Systems for Health Monitoring and Prognosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(1), 1-12. https://doi.org/10.1109/TSMCC.2009.2032660

3. **Hsu, Y., & Li, X.** (2022). Cyber-Physical Systems: Architecture, Design, and Applications. *Springer.* https://doi.org/10.1007/978-3-030-66110-0

4. **Barton, C. R., & Hong, Y.** (2018). The Role of Cyber-Physical Systems in Smart Manufacturing. *International Journal of Computer Integrated Manufacturing*, 31(12), 1246-1261. https://doi.org/10.1080/0951192X.2018.1496427

5. **He, H., & Wu, S.** (2021). Security and Privacy Challenges in Cyber-Physical Systems. *IEEE Access*, 9, 16867-16878. https://doi.org/10.1109/ACCESS.2021.3054342

6. **Zhang, J., & Lu, Y.** (2018). Data Security and Privacy Protection Issues in Cyber-Physical Systems. *International Journal of Information and Computer Security*, 10(4), 332-344. https://doi.org/10.1504/IJICS.2018.091373

7. **Sokolov, D. S., & Ma, J.** (2019). Edge Computing for Cyber-Physical Systems: Opportunities and Challenges. *IEEE Transactions on Industrial Informatics*, 15(1), 53-62. https://doi.org/10.1109/TII.2018.2877313

8. **Gao, L., & Zhang, X.** (2016). Cyber-Physical Systems: A New Generation of Embedded Systems and Cyber-Physical Systems. *Journal of Computing and Information Science in Engineering*, 16(4), 44-50. https://doi.org/10.1115/1.4034316

9. **Zhao, W., & Li, M.** (2020). Internet of Things for Smart Cities: Opportunities and Challenges. *IEEE Internet of Things Journal*, 7(6), 5492-5501. https://doi.org/10.1109/JIOT.2019.2958095

10. **Khan, R. H., & Zaman, A.** (2018). Energy-Efficient IoT in Cyber-Physical Systems: Concepts and Challenges. *Journal of Internet Services and Applications*, 9(1), 1-12. https://doi.org/10.1186/s13174-018-0089-4

11. **Cao, Y., & Zhong, X.** (2019). Challenges in Communication Protocols for Industrial Cyber-Physical Systems. *IEEE Transactions on Industrial Informatics*, 16(1), 129-137. https://doi.org/10.1109/TII.2018.2875950

12. **Grigoryev, D. S., & Kasyanov, I. V.** (2022). Cloud-Edge Computing for Cyber-Physical Systems: Architecture, Challenges, and Applications. *Journal of Cloud Computing: Advances, Systems and Applications*, 11(2), 1-20. https://doi.org/10.1186/s13677-022-00360-0

13. **Gerst, S. D., & Howard, B. D.** (2017). Interoperability Challenges in Cyber-Physical Systems. *Proceedings of the 2017 IEEE International Conference on Cyber-Physical Systems* (ICCPS), 1-8. https://doi.org/10.1109/ICCPS.2017.7583490

14. **Jia, W., & Yang, Z.** (2021). Blockchain for Secure and Privacy-Preserving Cyber-Physical Systems. *IEEE Transactions on Industrial Electronics*, 68(9), 8606-8615. https://doi.org/10.1109/TIE.2020.2986964

15. **Liu, Z., & Xu, J.** (2019). A Survey on Cyber-Physical Systems and the Internet of Things. *International Journal of Computer Applications*, 42(4), 54-60. https://doi.org/10.5120/ijca2019918580

16. **Zhang, L., & Zheng, X.** (2020). Real-Time Data Analytics in Cyber-Physical Systems. *IEEE Transactions on Industrial Informatics*, 17(5), 3150-3160. https://doi.org/10.1109/TII.2020.2973228