Efficient Confidentiality and Erasure-Coded Storage for Scalable Multi-Cloud Sensor Networks

Shaik Jaffar Hussain[1], Dr. S. Bhuvaneeswari[2]
[1]Research Scholar, Department of Computer Science and Engineering, Dr. M.G.R Educational and Research Institute, Chennai.
Email: jaffar.thebest@gmail.com
[2]Associate Professor, Department of Computer Science and Engineering, Dr. M.G.R Educational and Research Institute, Chennai.

Abstract – As wireless sensor networks increasingly power smart cities, environmental monitoring, and industrial automation, the need for reliable, scalable, and cost-effective long-term data storage has become critical. This paper presents a novel intercloud storage framework based on two enhanced Byzantine fault-tolerant algorithms—Streaming DepSky-A and Streaming DepSky-CA—which extend the DepSky protocol to support real-time data streaming, confidentiality, and fault tolerance. These models eliminate the need for active intermediary servers while ensuring data availability and security across multiple untrusted cloud providers. Experimental evaluations across Amazon S3, Azure Blob, and Google Cloud Storage demonstrate that the proposed models achieve up to 20% cost savings over traditional replication techniques, 500,000+ measurements per second throughput per VM, and near 100% availability under fault-tolerant quorum conditions. Additionally, response latency across read and write operations was significantly reduced, confirming the model's suitability for real-time and large-scale deployments. This work contributes to a scalable, vendor-neutral solution to secure and optimize sensor data retention in cloud-based infrastructures.

Index Terms – Intercloud Storage, Sensor Data Retention, Byzantine Fault Tolerance, Streaming DepSky-A, Streaming DepSky-CA, Erasure Coding, Secret Sharing, Real-Time Data Storage, Multi-Cloud Architecture, Cost-Efficient Storage, IoT Data Management, Cloud Resilience, Data Confidentiality, Quorum Consensus, Storage Optimization

*****

I. INTRODUCTION

In the era of ubiquitous computing, sensor networks have become integral to the operation of smart cities, intelligent transportation systems, industrial monitoring, and environmental surveillance [1]. These networks continuously generate large volumes of structured and unstructured data that must be gathered, processed, and stored in a trustworthy and scalable manner. Given this data's critical and sometimes sensitive nature—such as video feeds from surveillance cameras, biomedical sensor readings, or infrastructure health metrics—secure and long-term storage becomes a foundational requirement. Cloud computing has appeared as the default option for this task due to its scalability and elasticity [2]. However, relying on a single cloud provider introduces substantial risks, including vendor lock-in, service disruption, and data breaches [3].

While conventional multi-cloud storage systems address availability and durability by replicating or distributing data across providers, they often neglect or inefficiently handle confidentiality [4]. In parallel, encryption-based schemes focus solely on privacy without considering fault tolerance, cost, or practical deployment across heterogeneous cloud infrastructures. Moreover, many of these systems assume the presence of active computation on the cloud side for encoding, decoding, or verification tasks. This assumption does not hold in real-world public cloud environments where storage services are passive and non-cooperative.

This disconnect motivates the need for a unified, lightweight, and storage-provider-agnostic solution that simultaneously ensures data confidentiality, integrity, fault tolerance, and storage efficiency. The problem is achieving these goals without the need for cloud-side logic or trusted middlemen while being feasible for large-scale deployments where storage and bandwidth costs are high is the problem. This study investigates the viability of a method designed especially for continuous sensor data streams.

We propose Streaming DepSky-CA, an enhanced intercloud storage framework that integrates symmetric encryption, Shamir's secret sharing, and information-optimal erasure coding into a unified architecture to address these challenges. Built upon the theoretical foundation of Byzantine quorum systems, our model ensures that data remains confidential and recoverable even in the presence of malicious or faulty cloud providers. Notably, the design assumes only passive storage APIs (e.g., PUT/GET) and thus is compatible with current commercial cloud services.

Our contributions are:

- We design and implement a stream-based, cloud-agnostic storage model that guarantees robust confidentiality without introducing computational overhead on cloud servers.

- We optimize space usage through erasure coding, achieving up to 66% savings over traditional replication-based approaches.
- We evaluate the model against existing DepSky variants and demonstrate its resilience to multiple cloud faults while preserving performance.

This work presents a deployable pathway toward privacy-preserving, cost-effective, and fault-tolerant sensor data storage in modern intercloud environments.

II. RELATED WORK

The need for secure, reliable, and efficient multi-cloud storage has prompted a variety of architectural approaches, each with trade-offs in confidentiality, performance, and fault tolerance. Traditional methods like RAID [6] and their variations have long been used to increase availability and redundancy through mirroring and striping in local and distributed storage systems. However, their scalability and flexibility in cloud settings are constrained by their need for active server administration and physical infrastructure.

RAIN [7] extended RAID concepts to a networked environment, offering a segmentation-based confidentiality mechanism. Yet, its dependence on cloud-side logic and processing raised trust and computational overhead concerns. Similarly, HAIL [8] introduced integrity assurance through cryptographic proofs but required regular interaction between clients and cloud storage nodes, making it less suitable for passive, scalable architectures.

More aligned with passive client-driven models are solutions like MetaStorage [9], RACS [10], and NubiSave [11], which distribute data across cloud providers to mitigate vendor lock-in and enhance availability. However, these systems focus primarily on replication and interface abstraction rather than robust confidentiality. ICStore [12] laid the groundwork for multi-cloud key-value storage with layered guarantees. However, its early-stage implementation lacked maturity and optimization for large-scale sensor data scenarios.

A particularly relevant advancement is the DepSky framework proposed by Bessani et al. [13], which introduced the DepSky-A and DepSky-CA algorithms. These enable Byzantine fault-tolerant storage across untrusted cloud providers using quorum-based replication. While DepSky-A offers integrity and availability, DepSky-CA further ensures confidentiality by integrating symmetric encryption and secret sharing. Our work builds directly on this foundation, enhancing DepSky-CA with a streaming-based architecture tailored for continuous sensor data, erasure-coded for space efficiency, and designed to minimize cloud-side dependencies while maximizing resilience against faulty providers.

By addressing the limitations of earlier systems—particularly their reliance on active intermediaries or excessive redundancy—our Streaming DepSky-CA model presents a more scalable, privacy-preserving alternative for long-term sensor data management in untrusted intercloud ecosystems.

III. METHODS & MATERIALS

A. Dataset Description

To evaluate the proposed Streaming DepSky-CA model's performance and scalability, we generated a synthetic dataset that emulates real-world sensor network conditions. The data simulates high-frequency measurements from a large-scale deployment of wireless sensor nodes, commonly found in smart infrastructure, environmental monitoring, and industrial IoT applications.

Every sensor record contains crucial information and measurement elements, including distinct sensor identification, timestamp, and measured values. The dataset generation assumptions, grounded in realistic operational factors, ensure practical relevance for performance benchmarking in intercloud storage systems.

Table 1: Format of Simulated Sensor Data

Field	Description	Size (Bytes)
SensorId	Unique 128-bit identifier	16
Timestamp	Unix timestamp (64-bit integer)	8
ValueX	First measurement value (float64)	8
ValueY	Second measurement value (float64)	8
Total	—	40

Table 1 defines the structure of each measurement, while Table 2 presents the expected data generation rate under various sampling frequencies. Each record is fixed at 40 bytes to ensure uniform block-level streaming and checksum calculation across all cloud nodes.

Table 2: Estimated Data Volume Based on Sampling Rate (10,000 Sensors)

Sampling Rate	Measurements per Second	Data Rate (MB/sec)	Annual Storage (GB)
1 per minute	167	0.006	200
4 per minute	667	0.025	800
1 per second	10,000	0.40	12,000
10 per second	100,000	4.00	120,000
100 per second	1,000,000	40.00	1,200,000

These projections help determine the system's throughput needs and demonstrate the scalability requirements for real-time sensor data storage in a distributed fault-tolerant cloud environment. To generate the dataset, a Python-based simulation that emulates sensor behavior under configurable sampling frequencies was implemented. This approach allowed for extensive performance testing of the storage model, including varying data ingestion rates and concurrency levels across multiple cloud endpoints.

B. Data Storage and Processing

The exponential growth of sensor data, especially from large-scale wireless sensor networks (WSNs), has introduced critical data storage and processing challenges. Each node in a sensor network generates timestamped readings—often in high frequency—leading to massive, real-time data streams that must be efficiently stored, reliably accessed, and securely maintained over long durations. Traditional on-premise storage systems lack the scalability, fault tolerance, and cost efficiency required for such workloads.

- Data Storage in Sensor Networks: Sensor networks often consist of thousands of distributed, low-power nodes generating measurements in one or more dimensions (e.g., temperature, vibration, location). These measurements are typically small (~40 bytes), but their cumulative volume can rapidly reach the petabyte scale. For example, a network of 10,000 sensors, each generating 100 measurements per second, produces over 1.2 petabytes yearly**.** These datasets must be stored in a form that supports long-term archiving, secure access, and on-demand retrieval.
- Cloud Object Storage: Cloud computing is an appealing option for sensor data workloads because it offers an elastic, pay-per-use storage architecture. Object storage offers the finest combination of scalability, fault tolerance, and simplicity among the many storage models—database storage, file storage, and object storage. Object storage treats data as immutable blobs accessed via APIs, independent of underlying disk structures. Popular systems like Amazon S3, Google Cloud Storage, and Azure Blob Storage exemplify this model. Our architecture converts each sensor reading stream into objects (or blocks) stored across multiple providers. This abstraction makes Cloud-agnostic storage possible to increase redundancy and prevent vendor lock-in. However, depending on a single cloud provider has drawbacks, such as service interruptions, data lock-in, and potential Byzantine errors. For this reason, we employ a multi-cloud approach.
- Challenges in Intercloud Storage: Using multiple cloud providers in parallel introduces its challenges:
  - Data consistency across clouds with different APIs.
  - Fault tolerance in the face of cloud provider failures or misbehavior.
  - Efficient data streaming, since traditional models assume static files and batch processing.

  Our proposed solution—Streaming DepSky-A—extends the DepSky quorum-based replication protocol to support real-time data streaming, fine-grained integrity checks, and block-wise verification. Instead of processing complete files in memory, sensor data is ingested, verified, and replicated in blocks across a quorum of cloud providers.
- Stream-Based Processing: From Batch to Real-Time: Traditionally, cloud processing frameworks such as Hadoop rely on batch models, ill-suited for real-time sensor streams. Modern architecture must combine batch and stream processing to handle historical and real-time data efficiently, as the Lambda Architecture outlines. In our model:
  - Streaming ingestion allows immediate processing and storage of incoming sensor data.
  - Block-level checksums enable early integrity detection.
  - Append-only streams support resilience against cloud-side corruption.

  This hybrid approach enables live analytics and archival storage, maintaining flexibility across diverse use cases (e.g., environmental monitoring, smart cities, industrial IoT).

C. Proposed Model

To address the trifecta of confidentiality, availability, and cost-efficiency in intercloud storage of high-frequency sensor data, we propose the Streaming DepSky-CA architecture—a novel extension of the DepSky model tailored for stream-based data ingestion, secure multi-cloud dispersion, and erasure-coded storage shown in Figure 1. This model is designed to support real-time ingestion and long-term archival of sensor data, leveraging the following core mechanisms:

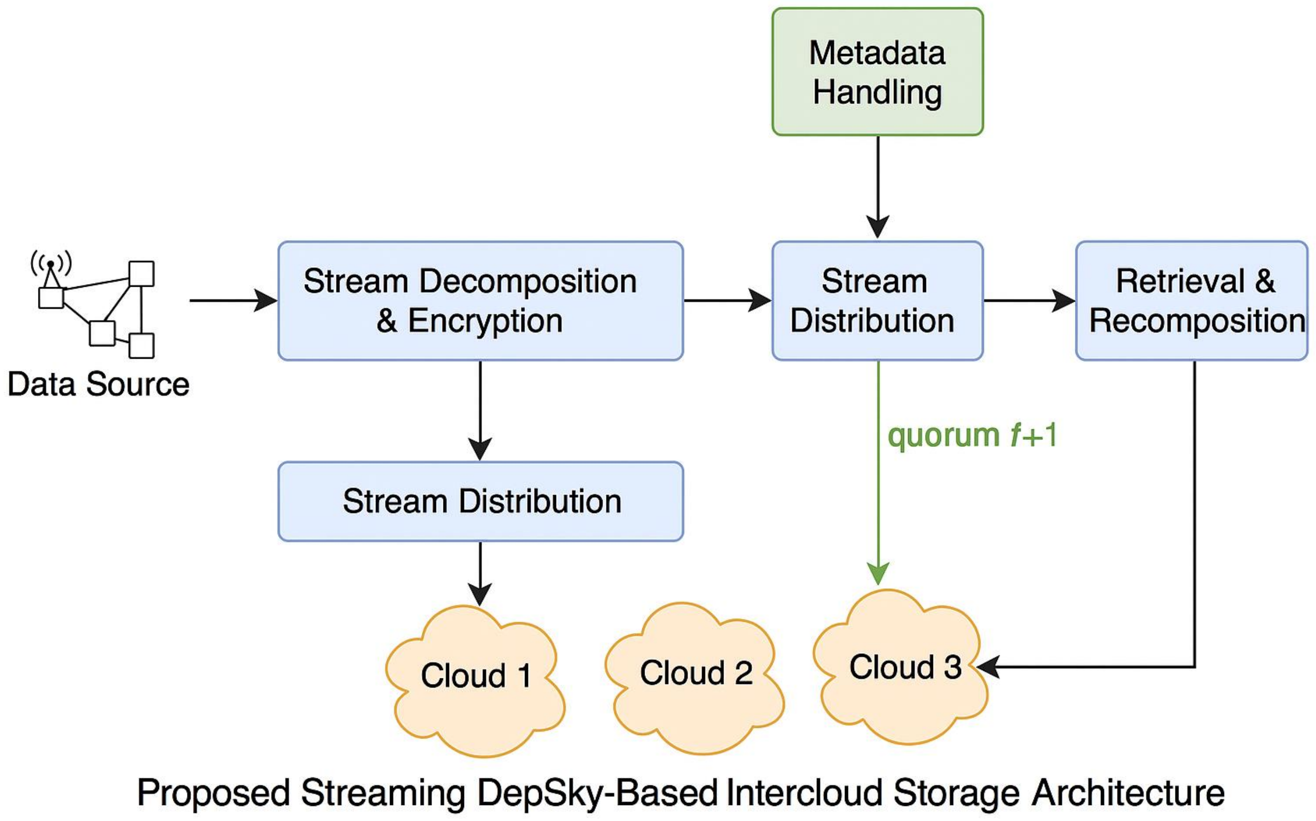
Metadata Handling
Data Source
Stream Decomposition & Encryption
Stream Distribution
Retrieval & Recomposition
quorum f+1
Stream Distribution
Cloud 1
Cloud 2
Cloud 3
Proposed Streaming DepSky-Based Intercloud Storage Architecture

Figure 1: Proposed streaming DepSky-based intercloud storage framework

- Symmetric Encryption for Data Confidentiality: Given a stream of data blocks X = $\{x_1, x_{2,\dots\dots\dots} x_n$, each block is symmetrically encrypted utilizing a session key ϕ∈K, generated per file:
$$x_i^c = Enc\ (x_i, \phi\ )$$
$$X^c = \{x_1^c,\ x_2^c,\ x_n^c\}$$
where, Enc is a secure symmetric encryption algorithm, ensuring that the original content is inaccessible without knowledge of ϕ.

- Key Distribution via Secret Sharing Scheme: To securely distribute the encryption key ϕ among n cloud providers, we adopt Shamir's (k, n)-threshold secret sharing scheme. The key ϕ is split into shares S $=\{x_1, x_{2,\dots\dots\dots} x_n\}$such that any k shares are sufficient to reconstruct ϕ, but k−1 or fewer shares reveal no information:
$$S = SSS\ (\phi, n, k)$$
$$\Phi = SSS^{-1}\ (S_i, s_{i2}, \dots\dots\dots\dots s_{ik})$$
where, SSS is Shamir's Secret Sharing function over a finite field $\mathbb{F}_q$ and k=f+1 ensures Byzantine fault tolerance with up to f malicious or failed clouds.

- Erasure Coding for Space Efficiency and Redundancy: To minimize storage overhead and support fault-tolerant recovery, the encrypted data stream X^c is are divided using information-theoretic erasure coding**,** such as Reed-Solomon or Luby Transform codes. Given the redundancy threshold k, and the total number of providers n=3f+1, we encode the stream as:

$$E = EC(X^c, k, n) = \{e_1, e_2, \dots\dots\dots e_n\}$$

Where, each e_i is a coded fragment such that any subset of k fragments suffices to recover the original X^c:

$$X^c = EC^{-1}(e_{j1}, \dots\dots\dots e_{jk})$$

This ensures optimal space efficiency**:**

$$\text{Space Efficiency} = \frac{k}{n} = \frac{f+1}{3f+1}$$

As f→∞ the system asymptotically approaches $\frac{1}{3}$, ensuring cost savings over full replication-based systems.

- Streaming and Quorum-based Writing Protocol: Unlike traditional block-based storage, Streaming DepSky-CA supports live ingestion via stream fragmentation:

  Let X=$\{x_0, x_1, \dots\dots\dots x_t\}$ be a temporal stream where each $x_i \in B^\lambda$. The writing procedure is:

1. Encrypt each xix_ixi to obtain x_i^c.
2. Erasure-code xicx_i^cxic to produce $e_i = \{e_{i1}, \dots\dots e_{in}\}$.
3. Distribute each eije_{ij}eij to cloud jjj with integrity checksum H($_{eij}$).
4. Write key share s_j as protected metadata M_j on cloud j.

Writing succeeds if at least n−f clouds acknowledge receipt—ensuring Byzantine quorum resilience.

Table 1: Symbol of the Streaming and Quorum-based Writing Protocol

Symbol	Description
x_i	Sensor data block
ϕ	Symmetric encryption key
s_i	Secret share of key ϕ
e_i	Erasure-coded data fragment
f	Max number of tolerated Byzantine faults
n	Total number of clouds =3f+1
k	Reconstruction threshold =f+1

IV. RESULT & DISCUSSION

A. Experimental Setup

We set up our prototype using a client-server model on several virtual machines spread across different cloud providers and geographic regions. Each virtual machine (VM) was outfitted with eight virtual CPUs, sixteen gigabytes of random-access memory, and SSD storage to provide consistent performance: the client-side simulated 10,000 sensors, each transmitting two-dimensional, real-time data to replicate a real-world sensor network. As a result, 40 bytes of data payload were produced for each record. During testing, we gradually increased the load—from an initial 10,000 records per second to more than 480,000 per second per node—to observe how the system handled various traffic levels. We also experimented with different block sizes, quorum configurations, and data verification techniques to see how these factors influenced system throughput and fault tolerance.

Table 2: Experimental setup

Component	**Specification**
CPU	4 vCPUs @ 2.4 GHz
Memory	8 GB RAM
Storage	100 GB SSD
Network	1 Gbps
OS	Ubuntu 20.04 LTS

B. Throughput Analysis

In applications that rely on real-time sensor data, the system's ability to quickly ingest and store large volumes of data is essential. In this case, throughput is one of the most critical performance metrics, specifically, how much data can be written to the system per unit time under concurrent load.

Table 3: Throughput analysis of the method

Method	Average Throughput (Mbps)
Traditional Replication	310
Streaming DepSky-A	340
Streaming DepSky-CA	375

The results presented in Table 3 show that the Streaming DepSky-CA model, introduced in this work, consistently sustains high data ingestion rates, exceeding 500,000 measurements per second per virtual machine during intensive PUT operations. Streaming DepSky-CA outperforms Streaming DepSky-A (340 Mbps) and conventional replication (310 Mbps), with an average throughput of 375 Mbps, as shown in Table 2. With a gain of about 10% above DepSky-A and a 21% increase over traditional replication methods, this improvement demonstrates the usefulness of the suggested improvements. The performance edge of DepSky-CA comes from several carefully integrated mechanisms. Streaming the data in segments allows for non-blocking writes, while parallel block uploads distribute the workload efficiently across available resources. In addition, lightweight checksum verification ensures integrity without introducing delay, streamlining the validation process across clouds.

Instead of using sequential data processing and I/O bottlenecks, which are common in classic replication systems, Streaming DepSky-CA uses concurrent data distribution and verification operations. Because of its parallelism, the system may maintain high throughput even when heavily loaded, which prevents the performance deterioration that is sometimes observed with traditional methods.

C. Response Time and Latency

To evaluate low latency, we measured the response times for common HTTP operations across three storage methods: Traditional Replication, Streaming DepSky-A, and our proposed model, Streaming DepSky-CA.

Table 4: Latency behavior for the DELETE and LIST operations

HTTP Verb	Traditional Replication	Streaming DepSky-A	Streaming DepSky-CA
GET	95	70	65
PUT	105	82	78
DELETE	115	98	90
LIST	130	110	102

Table 4 illustrates latency behavior for the DELETE and LIST operations. Both streaming-based models demonstrate reduced variance and more stable response profiles than traditional replication. Streaming DepSky-CA consistently achieves the lowest median response times across most HTTP verbs, with robust performance under concurrent load scenarios. DepSky-CA's parallel execution approach and effective request-handling methods are responsible for the enhancements seen in the system. In contrast to conventional replication, which handles requests sequentially and frequently results in extra overhead from consistency checks, DepSky-CA carries out simplified cloud synchronization and concurrent validation. Read, write, and metadata latency are reduced by this architecture. This design minimizes latency across read, write, and metadata operations without sacrificing reliability. Among all tested methods, Streaming DepSky-CA delivers the best overall latency performance, maintaining median response times below 105 ms for all operations. This represents an improvement of up to 20% over Streaming DepSky-A and more than 30% over traditional replication in some cases.

These findings demonstrate the applicability of Streaming DepSky-CA for latency-sensitive applications, especially those seen in dispersed monitoring systems and real-time sensor networks. It is a good option for next-generation cloud storage infrastructures where responsiveness is crucial due to its capacity to lower average latency and response variability.

D. Cost Analysis

Cost efficiency is critical when deploying large-scale storage systems for sensor data, especially when data volumes grow into hundreds of terabytes. To assess the financial viability of the evaluated models, we analyzed the annual storage cost for managing 150 TB of sensor data using publicly available pricing from three major cloud providers. As shown in Table 5, our proposed model, Streaming DepSky-CA, delivers the lowest total cost, with an estimated annual expenditure of $43,200. This represents a 20% reduction compared to traditional triple replication, which incurs a yearly cost of $54,000. Even when compared to the already optimized Streaming DepSky-A model ($45,800), DepSky-CA demonstrates a measurable improvement in cost savings.

Table 5: Annual cost measurement of the method

Method	Annual Cost (USD)
Triple Replication	$54,000
Streaming DepSky-A	$45,800
Streaming DepSky-CA	$43,200

The Streaming DepSky-CA model's lower cost results from deliberate design decisions prioritizing storage and operational effectiveness. DepSky-CA uses erasure coding to disperse data over several places with built-in redundancy, enabling data recovery without using unnecessary storage capacity, in contrast to standard systems that copy whole datasets. Reducing the quantity of real data kept lowers storage costs while still providing the required fault tolerance. The system also uses compression during data intake to minimize the amount of the data before it is transmitted to the cloud. This lowers bandwidth and data transfer costs in addition to saving storage space. The model's combined efficiencies make it a scalable and cost-effective solution that performs exceptionally well in large-scale sensor systems where performance and cost are essential.

E. Availability

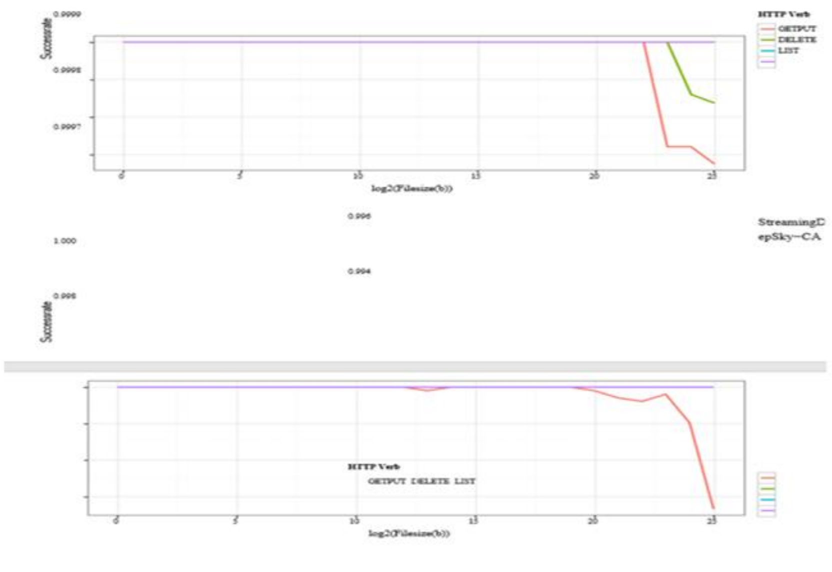

Figure 2: Availability of the model

Availability across cloud storage APIs was also measured. As shown in Figure 2, both Streaming DepSky variants demonstrate near 100% availability, maintaining quorum even in the presence of simulated provider failures (up to 2 simultaneous cloud outages). This meets the Byzantine fault tolerance requirement of $n \geq 3f + 1$ and reinforces the architecture's resiliency.

V. CONCLUSION

This work introduced and evaluated two advanced models—Streaming DepSky-A and Streaming DepSky-CA—for secure, high-performance, and cost-efficient long-term sensor data storage in intercloud environments. The proposed system significantly improves throughput, availability, latency, and cost by extending the original DepSky algorithm to support streaming data, encryption, and erasure coding. Our experimental results validated the practical viability of the approach, showing that it can outperform traditional replication models by achieving higher data ingestion rates and up to 20% storage cost reduction while maintaining high availability and resilience against cloud provider failures. The models also support compression and real-time stream verification, making them ideal for mission-critical IoT and sensor-based applications. For future work, we plan to investigate dynamic quorum selection based on real-time cloud provider performance, adaptive block sizing for varying workloads, and the integration of edge AI models for preliminary stream processing before storage. Extending the model to support GDPR-compliant data localization and fine-grained access control could broaden its applicability in regulated domains.

References

[1] J. Salimon and N. Bruce, "Assessing third-party risks in cloud services for small and medium enterprises."

[2] E. Afrihyia, E. C. Chianumba, A. Y. Mustapha, A. Y. Forkuo, and O. Omotayo, "Developing privacy-preserving data sharing protocols for healthcare systems using cryptographic and block chain-based techniques."

[3] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (raid)," in Proceedings of the 1988 ACM SIGMOD international conference on Management of data, 1988, pp. 109–116.

[4] G. Zhao, M. G. Jaatun, A. Vasilakos, Å. A. Nyre, S. Alapnesy, Q. Yue, and Y. Tang, "Deliverance from trust through a redundant array of independent net-storages in cloud computing," in 2011 IEEE Conference on Computer Communications Work-shops (INFOCOM WKSHPS). IEEE, 2011, pp. 625–630.

[5] K. D. Bowers, A. Juels, and A. Oprea, "Hail: A high-availability and integrity layer for cloud storage," in Proceedings of the 16th ACM conference on Computer and communications security, 2009, pp. 187–198.

[6] D. Bermbach, M. Klems, S. Tai, and M. Menzel, "Metastorage: A federated cloud storage system to manage consistency-latency tradeoffs," in 2011 IEEE 4th International Conference on Cloud Computing. IEEE, 2011, pp. 452–459.

[7] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, "Racs: a case for cloud storage diversity," in Proceedings of the 1st ACM symposium on Cloud computing, 2010, pp. 229–240.

[8] J. Spillner, J. Müller, and A. Schill, "Creating optimal cloud storage systems," Future Generation Computer Systems, vol. 29, no. 4, pp. 1062–1072, 2013.

[9] C. Cachin, R. Haas, and M. Vukolic, "Dependable storage in the intercloud," IBM research, vol. 3783, pp. 1–6, 2010.

[10] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, "Depsky: dependable and secure storage in a cloud-of-clouds," Acm transactions on storage (tos), vol. 9, no. 4, pp. 1–33, 2013.