MINIO

Object-Level Encryption for Data at Scale

"AIStor gave us encryption at scale without slowing down clinical workflows. Key management is centralized, auditable, and operationally simple."

— Security Architect, Large Hospital System

Every object isolated.
Every key ephemeral.

- **Per-object encryption:** Each object gets its own unique key. Breach one object, lose one object. Not a volume. Not a dataset.
- **Inline with erasure coding:** Encryption and data protection happen in a single atomic operation. No staging buffers where plaintext waits.
- **Keys never hit disk:** Generated in memory, used, destroyed. No persistent key database to become an attack surface.
- **AIStor MinKMS built in:** Key management at billions of objects without degradation. No external KMS required to operate.
- **Your master keys stay yours:** Integrate with your existing KMS or HSM. MinIO never holds root secrets.
- **Blast radius by design:** Architecture limits exposure to a single object, not a namespace or tenant.
- Every encryption event, every key operation, every access decision logged and provable.

The Challenge: Security That Scales Is Security That Gets Used

Storage encryption that slows down workloads gets disabled. Teams inherit bolt-on solutions that can't keep up with ingest rates, and they turn it off. The security team finds out at the audit. Meanwhile, data breaches cost $4.44 million on average globally, $10.22 million in the United States. The gaps are predictable: drives go back for RMA with recoverable data because the key was still accessible. Administrators can read plaintext because nothing separates storage access from data access. One compromised key exposes everything that shared it. These aren't edge cases. They're the norm.

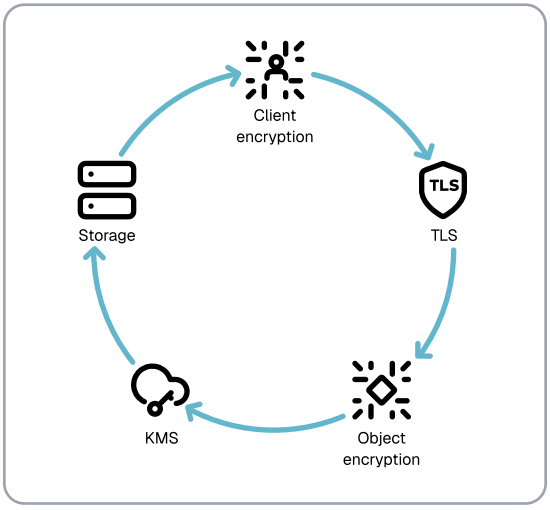
Enterprise-Grade Encryption

The encryption problems described above share a common root: legacy architectures treat encryption as a layer added after the fact, not a core part of how data moves and persists. AIStor takes the opposite approach. Encryption is built into the data path itself, executed inline with erasure coding, using per-object keys that isolate blast radius by design. The result is encryption that runs at ingest speed, scales to exabytes, and never exposes plaintext to disk or staging buffers.

Inline Encryption. When an object is written, AIStor encrypts it simultaneously with erasure coding in a single atomic operation. By the time the write is acknowledged, data is already encrypted and distributed across the cluster. There is no staging step, no buffer window where plaintext sits waiting.

All traffic moves over TLS 1.3 (TLS 1.2 fallback) with strong cipher suites enforced by default. AES-256-GCM encryption streams inline regardless of object size, with SIMD hardware acceleration keeping encryption at wire speed. This is how encryption keeps up with ingest rates instead of falling behind them. On read, encrypted objects are retrieved with their wrapped keys, AIStor calls the KMS to unwrap, and streams decryption back over TLS. Multipart uploads encrypt each part independently and validate authentication tags. Corrupted tags fail fast with clear errors rather than returning bad data.

Per-Object Keys. Each object gets its own unique encryption key, wrapped by your KMS-managed master key. If one key is compromised, one object is exposed. Compare that to whole-drive or volume-level encryption where a single key protects everything on that storage, and a single breach exposes it all. The wrapped key travels with the object metadata, so decryption requires only the object and KMS access. There is no central key database to query, no coordination overhead, no single point of failure.



Key Management

Encryption is only as strong as the key management behind it. AIStor integrates with the KMS infrastructure you already run and includes AIStor MinKMS for organizations that need a purpose-built solution for object-scale workloads.

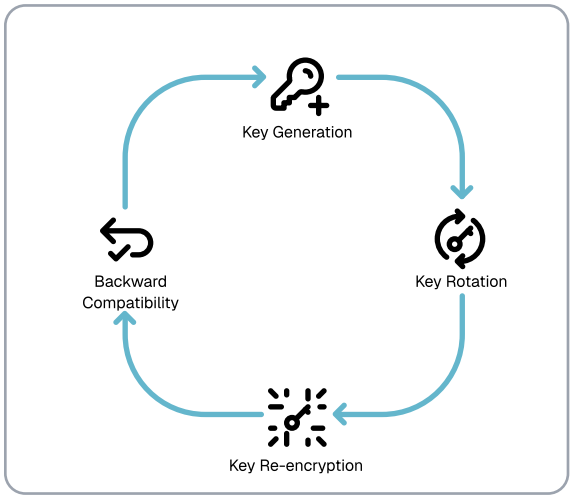
KMS Integration. AIStor works with AWS KMS, HashiCorp Vault, Azure Key Vault, Google Cloud KMS, and AIStor MinKMS. Three server-side encryption modes support different trust models. SSE-KMS provides granular per-bucket or per-object key control for regulated workloads. SSE-S3 simplifies management with a single key across the cluster. SSE-C puts the application in full control, supplying the key per request with AIStor never storing it. Client-side encryption is also supported for data that must remain unreadable outside the application.

	SSE-S3 Server-Managed Keys	SSE-KMS Customer KMS Keys	SSE-C Customer Provided Keys
Key Location	MINIO AISTOR	External KMS	Client-Side Only
Control Level	Basic	Moderate	Full
Ease of use	Simplest	Moderate	Complex
Ops Overhead	Lowest	Moderate	Highest
Risk Profile	Low Risk	KMS Dependecy	Lost Key=Lost Data

AIStor MinKMS. Traditional KMS and HSM products were built for thousands of keys, not billions. When every object has its own encryption key, legacy key management becomes the bottleneck. MinKMS is a high-performance, distributed KMS designed specifically for object-scale workloads. It deploys separately from the AIStor object store and scales horizontally for high availability. Synchronous replication across cluster nodes ensures each node maintains a full replica of the encrypted key database. The consensus model prioritizes read availability: encryption, decryption, and key derivation require only one available node. Administrative operations like key creation require all nodes. Cryptographic operations continue even during partial cluster failures.

Key Hierarchy. Your Customer Managed Key (CMK) lives inside your KMS or HSM and never leaves it. AIStor uses the CMK to derive Data Encryption Keys (DEKs) for each object. Each DEK is wrapped and stored with the object metadata. Decryption requires the CMK to unwrap the DEK. Delete the CMK and all related data becomes permanently unreadable. This separation means you control the keys while AIStor handles the scale.

HSM Integration. MinKMS optionally integrates with Hardware Security Modules to seal and unseal root encryption keys. The root key exists only within the HSM boundary, never in software. This architecture supports FIPS 140-2 validation and NIST guidelines for organizations with strict compliance requirements.



Live Key Rotation. Key rotation happens without maintenance windows or service interruption. Administrators define rotation policies by object age, tags, metadata, or KMS key ID. AIStor processes matching objects in the background, re-wrapping each object's DEK with the new master key. The data payload is untouched, only the key envelope changes. Objects encrypted with older key versions remain readable until re-wrapped. TLS certificates also reload live without restarting services.

Audit Logging and Access Control. AIStor enforces IAM policies controlling who can request KMS operations, for which key names, and under what conditions. Every key operation is recorded with identity, timestamp, and outcome. Administrative KMS API calls generate audit events. Failed decryption attempts are logged with the calling principal and failure reason. This is how teams prove control to auditors, not just claim it.

Traditional Compliance Architecture vs MinIO AIStor

	Traditional Storage Encryption	MinIO AIStor
Encryption Granularity	Whole-drive or volume-level	Per-object unique keys
Key Management Scale	Thousands of keys	Billions of keys
Encryption Timing	Staged, post-write	Inline with erasure coding
Performance Impact	Measurable degradation	Near-zero (SIMD accelerated)
Key Rotation	Maintenance windows required	Live, zero downtime
Compromise Blast Radius	Entire volume or dataset	Single object

Why MinIO AIStor

AIStor delivers encryption that works at the scale modern enterprises require. Per-object keys eliminate broad compromise scenarios. Inline encryption removes staging vulnerabilities. MinKMS handles billions of keys without becoming a bottleneck. Key rotation and certificate updates happen live. Every key event is logged with identity, timestamp, and outcome.

Ready to see it in action?

Visit min.io to learn more. Download AIStor and test it yourself. Request a demo to see how AIStor's encryption works with your infrastructure and compliance requirements.