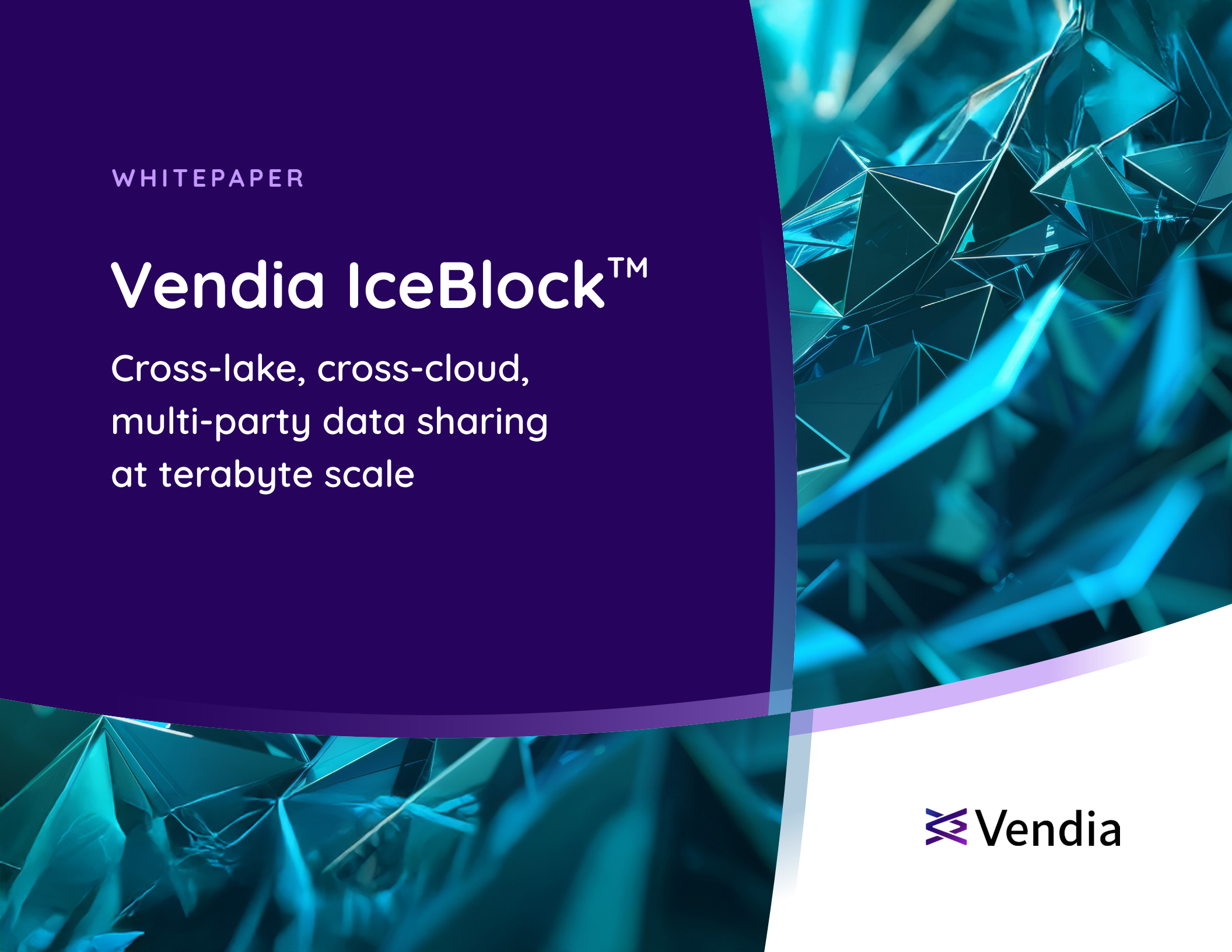
WHITEPAPER

Vendia IceBlock™

Cross-lake, cross-cloud, multi-party data sharing at terabyte scale



Vendia

Apache Iceberg is an increasingly popular open-source format for representing and sharing large datasets. **Vendia IceBlock™** enables Iceberg data to be shared reliably across different parties, clouds, and data lakes while preserving security, trust, and transparency in the underlying data. It extends Vendia's existing solutions for trusted, auditable operational data sharing in multi-party environments to terabyte-scale datasets.

Vendia IceBlock enables real-time distributed data clean rooms, distributed data fabrics, heterogeneous data sharing, and other use cases while improving governance, privacy protection and auditability of shared data at effectively unlimited scale. IceBlock is compatible with popular cloud object stores, including Amazon S3 and S3 Table Buckets, as well as Azure Blob Store.

Background

A primer on Apache Iceberg

Apache Iceberg, often referred to simply as "Iceberg," is a popular, open-source format for representing and managing large data sets. Originally devised for analytical and BI use cases, it's also found favor in AI/ML, data sharing and monetization, and a host of other applications.

Iceberg enables efficient storage and safe, incremental updates of large datasets. While adaptable, it's widely used with cloud object stores like Amazon S3, making it popular for data lake environments.

Iceberg's open-source nature and vendor-neutrality appeal to both developers and companies as an alternative to proprietary, "internal" database and data lake formats. This aligns with a broader industry trend toward "data neutrality," where data is stored on neutral platforms, such as public cloud object stores like Amazon S3. This approach offers greater flexibility and reduces vendor lock-in.

As data lakes, ETL processes, and AI/ML tools adopt this "Iceberg-tables-in-a-cloud-blob-store" approach as their preferred storage solution, this combination has become a de facto standard for organizations of all sizes across industries.

Understanding enterprise blockchain

Enterprise blockchains are private, permissioned ledgers ensuring data integrity and synchronization across authorized participants. Ensuring the correct dataset version reaches intended destinations across diverse environments—multiple clouds, geographies, systems, and legal entities—presents a significant challenge as businesses share vast amounts of data.

This information is the basis for compliance regimes such as GDPR, CCPA/Prop 24, open banking, antitrust provisions, and more. Having an auditable trace in which all parties can definitively prove "who knew what and when" dramatically simplifies the challenges of sharing sensitive data, enabling all parties to use data with confidence in its accuracy and fidelity.

Historically, distributed ledger technology excelled at sharing small, highly trusted data. Conversely, large datasets, often utilizing formats like Iceberg, lacked the inherent trust and traceability offered by ledgers. Vendia's IceBlock platform resolves this dichotomy, extending the core benefits of a distributed ledger—trust, security, and traceability—to terabyte-scale data while seamlessly integrating with the Iceberg data format.

The challenges of sharing large datasets

Data-sharing features in data lakes, such as Snowflake's table-sharing capabilities, have revolutionized how data operators and companies approach large-scale data collaboration, simplifying the process of making vast quantities of data trivially accessible across a company's many divisions, functions, and business partners.

But what happens when the two parties sharing data use different clouds, data centers, or data lakes? Even when both parties share a common technology stack and set of vendors, remembering which subset of which version of which data went to which part(ies) can be a frustrating and complicated exercise, and only grows worse when the data contains PII, PHI, sovereignty/locale restrictions, or is subject to other regulatory or legal constraints. Tracking these details at scale and in real time becomes a daunting task, even for the largest and most IT-savvy companies.

Vendia IceBlock extends the easy data-sharing capabilities on Snowflake and similar platforms to a broader spectrum of use cases, enabling seamless data movement across clouds, data lakes, and geographic boundaries. This ensures data lineage, access control, and privacy compliance are maintained throughout the data's journey, and establishes an immutable and auditable record of all data-sharing activities—i.e., who shared what with whom.

Vendia IceBlock brings the security and reliability traditionally associated with ledgering small data payloads to terabyte-scale datasets, all while preserving ease of use and optimal performance. Importantly, Vendia IceBlock also:

- Retains the open, standards-based format of Iceberg datasets, including the ability to leverage all existing Iceberg-conformant tools, features, and services. Iceberg files adhere to industry standards and are fully compatible with major cloud object stores, including Amazon S3 and Azure Blob Storage.
- Enables sharing large, terabyte-sized datasets across different lakes, clouds, geographic regions, and parties, as well as between on-prem and cloud environments.
- Adds the security, fidelity, immutability, and irrefutability of blockchains without adding an appreciable amount of cost or latency or reducing the availability or scalability of large dataset sharing solutions.
- Integrates with existing on-prem systems through JDBC/ODBC connectors or SFTP.

How Vendia IceBlock works: Hierarchical file digests

When a file is stored in a cloud object service, such as Amazon S3, how does the company storing the information know the resulting file is correct? That it hasn't been truncated or corrupted?

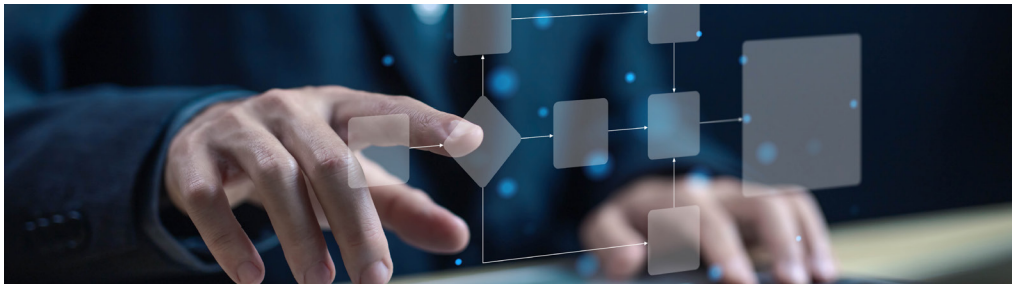
To ensure fidelity, services such as S3 compute a "digest" or "hash", which is a short string that summarizes the entire file's content. By comparing digest values, companies can efficiently verify a single file's integrity after uploading it, without needing to retain and compare the entire file's content. For files stored in S3, this hash uses an algorithm known as "MD5". The file owner can independently calculate the MD5 and then compare it to the hash generated by S3. This comparison verifies that the file was successfully delivered and that its integrity remains intact—initially and over time.

Vendia IceBlock extends this "verification through hashes" capability from **one file** to **a collection of files** stored in Iceberg format. In Iceberg, the individual files making up a data set are already immutable: Instead of rewriting data files to update their content, new files are created to hold the updated or changed information.

This process also extends to directories. Instead of changing a directory when a file is updated (really, replaced), a new version of the directory is also created, and so on all the way up the hierarchy. This immutability is great news for placing Iceberg content in a ledger because it eliminates the concern of managing different versions of a file's content[1].

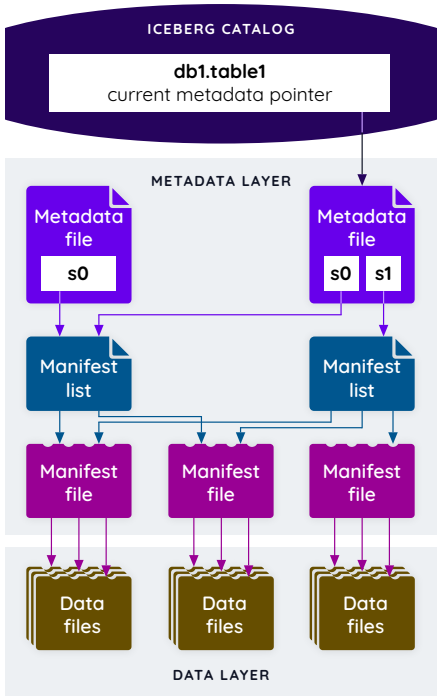
To extend S3's individual MD5 file hash approach, IceBlock also computes cumulative hashes at each directory level and then stores the final (root) hash in the data catalog recording Iceberg table information. In technical terms, it creates a Merkle tree of hashes isomorphic to the original Iceberg file collection's structure. This also means that Iceberg's support for incrementality—i.e., the ability to efficiently make updates to the data—naturally extends to IceBlock as well.

IceBlock focuses on the file layout of the data, designed to ensure that its original and subsequent versions can be shared with any number of parties, clouds, lakes, and geographies in a read-only manner. Like an MD5 hash, it also ensures that data hasn't drifted or been tampered with over time.



[1]Iceberg tools also add unique elements to file names to ensure that files don't accidentally collide with one another, avoiding both intentional and accidental versioning of a file's content. Essentially, any given file name is used only once, with its content remaining the same until it gets deleted.

IceBlock at a technical level: Merkle trees for Iceberg's storage layer



To understand how IceBlock puts Iceberg data on a blockchain, let's start with a quick review of how Iceberg storage works. Iceberg's storage layer consists of five key components:

1. **Catalog:** The top layer, typically employing optimistic locking when modifications occur in the underlying snapshot layer.
2. **Metadata file:** This file acts as a log of table data snapshots. Each data change results in a new metadata file with an appended snapshot.
3. **Manifest list:** Represents a snapshot of the table's data. It serves as the top level of a two-tiered directory structure.
4. **Manifest file:** The lower level of the directory hierarchy, which points to one or more data files.
5. **Data file:** Typically a Parquet-formatted collection of data.

Put another way: Catalogs reference snapshot lists.
Each snapshot points to a manifest list (upper directory), which in turn points to manifest files (lower directory). These manifest files point to the actual data files.

As mentioned, individual files are write-once throughout their lifetimes:

I. **Copy-on-write (COW) for metadata:** Metadata files, including manifest lists and manifest files, are always updated using the COW approach. This means that whenever a change occurs (e.g., new data addition, compaction), a new copy of the file is created instead of modifying the existing one.

II. **Data file handling:** Data files can be handled either with the COW approach or the merge-on-read (MOR) approach. In MOR, changes are recorded in delta files, which are applied on read to obtain the updated data. However, at the file level, MOR still involves creating new data files and updating metadata accordingly, effectively making it a form of COW at the file system level.

III. **Catalog updates:** The catalog, which resides outside the core Iceberg storage format, must be updated atomically whenever metadata files are modified. This atomicity enables transactional updates by "swapping the root" of the metadata hierarchy. This essentially changes the "pointer" to the current table version within the catalog.

On a more technical level, Iceberg's storage tier can be conceptualized as a structure-sharing forest of directed acyclic graphs (DAGs). Each modification to a data file (addition, deletion, or update) introduces a new path to the root (the catalog). However, Iceberg maximizes the reuse of existing structures.

Consider a simple scenario where a single row in the data is updated (using the COW approach for data files). In this hypothetical scenario with 100 data files, 10 manifest files, a single manifest list, and a single metadata file:

Data files	Manifest files	Manifest list	Metadata file	Catalog
99 data files remain unchanged; **1 new data file** containing the updated row is created.	9 manifest files remain unchanged; **1 new manifest file** is created to point to the newly added data file.	The existing manifest list is replaced with **a new one** that now points to the 9 unchanged manifest files and the newly created one.	The existing metadata file is replaced with **a new one.** This new metadata file adds a new snapshot to its list and points to the updated manifest list.	The catalog is updated to reference the latest version of the metadata file.

In this example, only 4 out of 112 files are actually replaced. Furthermore, since data files typically occupy significantly more space than metadata files, the majority of the data remains unchanged. This approach ensures efficient updates while strictly adhering to the principle of file immutability at all levels of the hierarchy.

From Iceberg to Merkle trees

Conceptually, a distributed ledger is a sequential record of data updates. Each update, known as a "block," is linked to its predecessor, forming a chain. To ensure data immutability, blockchains employ two key techniques:

1. **Merkle tree:** Each block includes a Merkle tree, a hierarchical hash structure that provides a digest of all data within the block. The root hash of this Merkle tree serves as the unique identifier for the block's contents.
2. **Hashing:** Each block also contains a copy of the preceding block's hash, along with a unique identifier.

These two techniques work in tandem to provide strong immutability guarantees:

- Any attempt to "rewrite history" or alter the order of blocks can be readily detected as the hash of the preceding block within each block will no longer match.
- Modifying the data within a block will inevitably change the data's hash. This will result in a different Merkle tree root hash, making the alteration easily detectable.

Figure 2 visually illustrates this approach by expanding the Merkle tree for block 1, clearly demonstrating how it protects the data contained within that block.

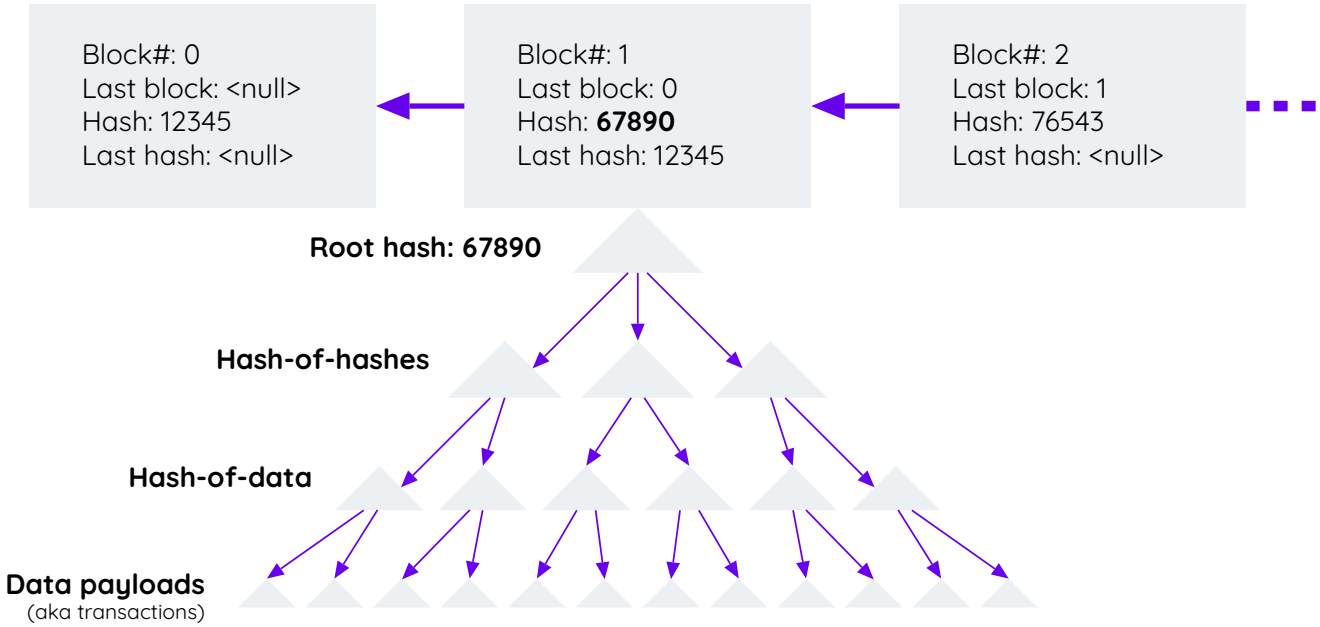


Fig. 2: How blockchains protect data using Merkle trees (digests-of-digests)

File hashes and the “downward only” trust model

The final component is the cloud object store itself. In the IceBlock model, “data” encompasses all Iceberg storage files, including both data files and the associated directory metadata files. These files are exclusively stored within a public cloud object storage service. For the sake of clarity and conciseness, we will focus on Amazon S3 throughout this discussion, deferring multi-cloud considerations to a later stage.

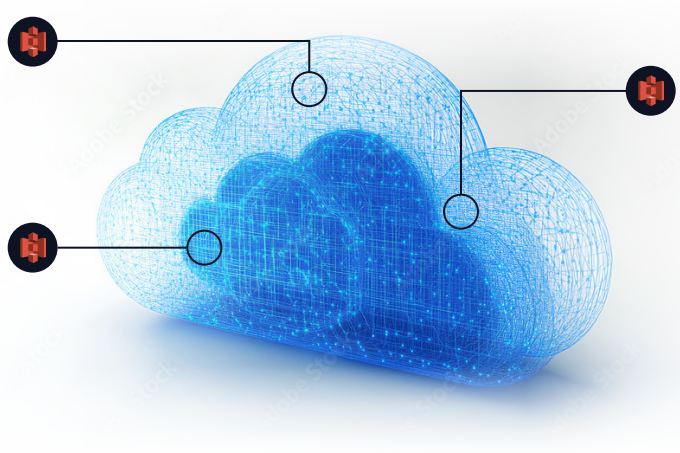
S3 provides several key features that are crucial for IceBlock:

Object immutability

Objects can be added, replaced, or deleted, but their content remains unchanged after upload.

Versioned buckets

Versioning can be enabled for buckets, ensuring that any attempt to replace a file creates a new version while preserving the original.



MD5 hashing

For files under the 5GB limit adhered to by Iceberg, S3 automatically calculates and stores an MD5 hash in the file’s metadata. This “etag” serves as a reliable hash for use within the Merkle tree.[2]

IceBlock, like most blockchains, relies on “downward” trust. This means participants don’t need to trust each other but instead trust the underlying infrastructure. In this example, all parties sharing data on the chain trust Amazon S3 (or equivalent cloud storage) to reliably store and retrieve files, maintain their integrity (including the etag), and adhere to its service-level agreements.

This “downward” trust extends beyond the public cloud provider to encompass chip manufacturers, network device manufacturers, etc. (i.e., all the way down to the “silicon and copper”).

[2]Note that this “etag” refers specifically to the MD5 hash generated by S3 and should not be confused with other uses of the term. While S3 does not calculate MD5 hashes for files larger than 5GB, this limitation has no impact on Iceberg or Iceblock, as they operate within a file size range that does not exceed this threshold.

Putting the IceBlock pieces together

With these pieces in place, IceBlock can be simply described as a “Merkle tree for Iceberg files stored in S3.” We construct a structure-sharing DAG, mirroring the Iceberg storage graph, but populated with hashes instead of file pointers. This forms a Merkle tree. The root of this tree can be integrated into the Iceberg catalog (if supported), stored alongside the Iceberg files in S3, or placed in any convenient location.

Platforms like Vendia automate this process by computing and storing the Iceberg table’s Merkle tree within a distributed ledger, facilitating easy sharing across parties, clouds, and data lakes. The efficiency of Iceberg’s structure-sharing mechanism directly translates to the Merkle tree recalculation process. When a new block is added, only the hashes of modified files need to be updated.

In our earlier example, where a single row was modified, only 4 new hashes would be required, instead of recalculating all 112 hashes. Unchanged nodes in the Merkle tree can be reused from previous blocks or simply point to earlier versions in the ledger.

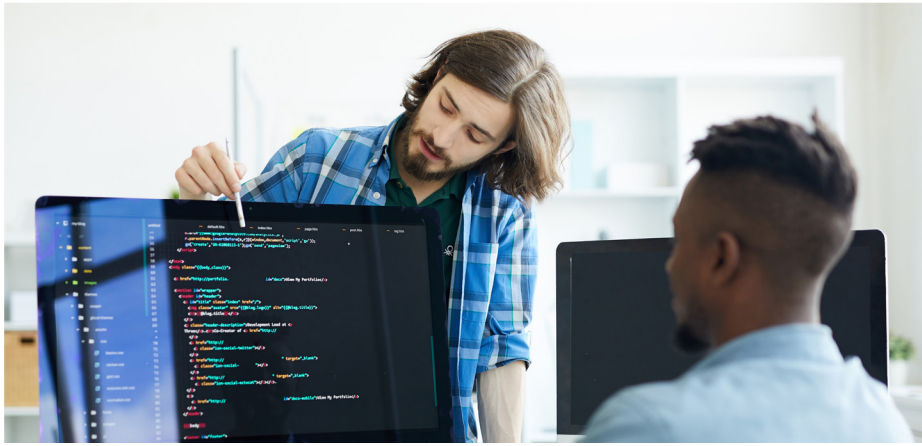
Merkle tree entries for manifest, manifest list, and metadata files contain two types of hashes:

1. The hash of the underlying Iceberg file.
2. A “hash of hashes” summarizing the hashes of all files referenced by the underlying file.

The Merkle tree can be stored within the block using a suitable format like JSON. Alternatively, it can be embedded directly within the blob storage layer through the following methods:

A. **Storing hashes in S3 object metadata:** This method requires modifying the existing files to include the Merkle tree hashes within their metadata.

B. **Creating “shadow” files:** This approach involves creating separate “shadow” files that contain the Merkle tree hashes. To ensure the integrity of both the data and the shadow files, additional hashes for the shadow files themselves must also be stored. This necessitates storing hashes for the shadow files to ensure their integrity alongside the underlying data.



Limitations and extensions

IceBlock is primarily designed for one-to-many, read-only data sharing, similar to the functionality of intra-platform solutions like Snowflake's data sharing. The difference is that IceBlock extends this capability to heterogeneous environments across different clouds and data lakes.

IceBlock preserves Iceberg's real-time, incremental update capabilities. This allows for virtually unlimited updates to be tracked efficiently, with the processing time and storage overhead proportional to the update size rather than the entire dataset.

Two or more parties can also set up bidirectional sharing to exchange information back and forth, provided the tables are kept separate. Editable (non-read-only) sharing presents greater challenges. Asymmetric data updates by one party or operational changes like table resharding can lead to inconsistencies in the file-level storage across different parties.

Extensions to IceBlock, as outlined in this document, can address some of these use cases by employing more selective hashing techniques instead of solely relying on the native MD5 file hashes provided by the underlying object storage system.

SaaS data sharing with Vendia

Creating data fabrics and sharing ecosystems, both within and across business boundaries, is one of the most critically important IT challenges today. With mission-critical imperatives like GenAI, ML, and BI at the top of nearly every CEO's priority list, getting the right data in and out of an organization has never been more timely or critical—thus, the ease with which this data can be shared, without losing trust, fidelity, or control, is equally crucial.

Businesses are increasingly sharing all types of information (e.g., structured, semi-structured, and unstructured) in more ways (e.g., APIs, SQL, SFTP, data lake sharing, etc.) at a time when regulations and privacy concerns are growing by the day. Yet the solution to this complexity isn't a return to the past problems of data silos and data islands. Rather, it's to retain and extend the ease of data sharing while bringing in all the necessary trust, control, security, and governance capabilities to the process.

Vendia IceBlock extends Vendia's existing ability to easily share operational and file data to effectively unlimited scale, allowing terabyte-sized datasets to be shared with blockchain-grade protections. Both materialized views and virtual sharing can be accomplished across "data canyons"—i.e., on-prem versus public cloud, different public clouds, multiple geographies, two or more parties or divisions, and more.

Vendia IceBlock extends support for governance, trust, and privacy-preserving data sharing to a new dimension of corporate data, unifying all forms of data sharing in a single, easy-to-use and API-driven platform. Use cases such as data monetization, real-time clean rooms, and TCO optimization between on-prem and public cloud have never been easier or safer to achieve.

To learn more about Vendia's innovative approach to making data sharing both trustworthy and simple, visit Vendia.com.

*Vendia IceBlock is a patent-pending technology of Vendia. Vendia and Vendia Iceberg are registered marks of the company.

About Vendia

Vendia is the future of collective data intelligence, combining smart APIs, databases, and distributed ledger technology inside a single platform. Vendia's data automation cloud makes it easy to share data inside and outside of the organization in real time and with full visibility, governance, and control. Companies such as BMW, Delta Airlines, Resolution Life Insurance, and Fannie Mae use Vendia to automate contextual and compliant data flows between any-to-any systems for a harmonized, accurate view of data that unlocks speed, innovation, and cost savings. Learn more about us at Vendia.com and #UnchainYourData with Vendia.

Vendia