Hermes

END-TO-END CRYPTOGRAPHIC ACCESS CONTROL FOR DISTRIBUTED SYSTEMS

The increasing number of infrastructure breaches, growing distrust in “walled garden” architectures, “perimeter security” and sufficiency of classical encryption schemes creates a demand for end-to-end encrypted online services.

Cryptographic framework Hermes provides the essential building blocks for creating end-to-end encrypted **zero-knowledge architectures**.

Hermes allows deploying end-to-end encrypted data exchange, sharing, and collaboration in your apps. It acts as a **protected data circulation layer** with cryptographic access control for your distributed application, with zero security risk of data exposure from servers and storage.

FAVORABLE USE

Hermes is useful for data protection within environments with complex yet strict security demands towards access policy.

The richer the data model is, the easier it is to protect it with Hermes and ensure deep integration of cryptography into the app’s security tools. Combined with proper key management, Hermes becomes a platform for secure data turnover on all stages of your application.

KEY BENEFITS

Data leaks don’t matter

The sensitive data is always encrypted during its lifecycle on all stages.

Cryptographic access control

Enforce group policies, integrate with PKI and AIM, while data is encrypted.

Zero Knowledge, Zero Trust

Prevents insider and outsider data leakage.

Developer-friendly API

Easy-to-manage encryption that reduces the security workload for engineers.

Secure distributed collaboration

Collaborate on data structures with cryptographically-checked permissions.

Fast and secure

Works without re-encrypting an excessive amount of data.

TYPICAL USE CASES

Security in data sharing applications: from shared storages to specialised cloud sharing and collaboration apps.

Automated sensitive data processing: enterprise record management, business process automation, business applications that operate with customers’ data at scale.

Data security in healthcare and fintech apps that build their processes around sensitive personal data (PII, PHI, EHR, FHIR).

Share documents across users with accountable logging, enforcing read/write policies while keeping data end-to-end encrypted.

Reaching compliance with privacy regulations through complete avoidance of processing unencrypted data in data lake, DWH.

COMPATIBILITY

Application languages: C/C++, Python, Go.

Core library: C/C++.

Server OS: CentOS, Debian, Ubuntu.

Storage: SQL/NoSQL databases, KV stores, filesystems.

LICENSE

AGPL 3.0 license for assessment and trial.

Commercial license for commercial usage: more integration languages, commercial support.

THREAT MODEL

Hermes was built to function in a very restrictive threat model:

- each system entity (user or service) can be compromised;
- data storage can be dumped;
- protected information can be partially leaked;
- data object model can be leaked;
- active attackers may be present in the communication channels.

SECURITY MODEL

Hermes can be mapped to the typical client-server architecture with the following security model:

- an **absence of a central point of security failure** (sensitive data being compromised);
- **no access** to both cryptographic keys and sensitive data in plain text for the server-side;
- **end-to-end authenticated encryption** between all the components (both server-side and client-side).

Hermes separates the cryptographic operations from the network-facing code: it reduces the potential attack surface, minimises the damage from discovered zero-day vulnerabilities, and simplifies the security audit of the system.

Even during the most extreme incidents (where all the server-side components are compromised), **most security guarantees are preserved** and the damage is limited (sensitive information appears in plain text only within one client's context).

OBJECT MODEL

A **recordset** in Hermes terminology means a list of records where at least one is non-protected (public, refence identifier) and the others are protected (private, sensitive data).

Hermes was built to work with the real-world entities: JSON documents, files, etc. However, traditional RDBMS row (or sub-set of cells in a row) can also be presented as a Hermes document (with either a private key or a row number being a public record).

ACCESS RIGHTS

Hermes operates with the **asymmetric key pairs** and their IDs as finite vessels of authentication. Each record within Hermes document is encrypted via a certain sequence of cryptographic transformations, depending on the rights assigned to public keys of users.

TRUST MODEL

Data owners are considered to be the sole Source of Truth for both data and access rights. It is enforced cryptographically.

SECURITY GUARANTEES

1. The compromise of a single entity from the system causes only limited damage.

2. All the sensitive information only appears in plain text in the system user/service's context exclusively.

3. Data is protected in a granular form (per record/document).

4. All communications are protected with end-to-end encryption.

5. Each data record is protected with a unique encryption key.

6. Each data record may have its own flexible access control policy.

LEARN MORE

Read the Scientific and Implementation papers about the mathematical and security model of Hermes and its public proof of concept implementation details

Visit the Hermes Open-Source GitHub repository to see the code and examples.

Read the Hermes documentation and use cases online.

Request a demo of Hermes-based platform to test end-to-end encryption, measure performance and try Hermes with different datastores