Hermes

End-to-end cryptographic access control for distributed systems.

COSSACK LABS

Essential benefits for:

- **Your clients:** secure data turnover on all stages.
- **Your developers**: easy-to-manage encryption that reduces the backend security workload, letting your engineers concentrate on their primary tasks.
- **Your product**: more user trust and lower risks of data breach.

Key features:

- Encryption layer for **end-to-end data exchange**, sharing, and access control in one software library.
- Granular distribution of access permissions in large data structures: documents, files, database cells.
- Compatibility with hundreds of data storage solutions, SQL/NoSQL databases, KV stores, filesystems.

Compatibility:

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

OS: CentOS, Debian, Ubuntu.

License:

- AGPL 3.0 license for assessment and trial;
- Commercial license for commercial usage: more integration languages, commercial support.

Overview

Due to 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.

Typical use cases

- Data security in exchange/sharing applications: from shared storages to specialised cloud sharing apps.
- Applications that automate sensitive data processing: enterprise record management, business process automation, business applications that operate with customers’ data at scale.
- Data security in healthcare and financial apps that build their processes around sensitive personal data.
- Secure data layer for reaching compliance with privacy regulations through complete avoidance of processing unencrypted data.

Favorable use

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

The richer object/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 can become a platform for secure data turnover on all stages of your application.

Hermes

End-to-end cryptographic access control for distributed systems.

COSSACK LABS

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,

yet data is protected.

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**: this reduces the potential attack surface, minimises the damage from discovered zero-day vulnerabilities, and simplifies the security audit of the system.

Even in the most extreme case of compromisation (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 the client's context).

Object model

Recordset in Hermes terminology means such 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 **real-world entities**: JSON documents, files, etc. However, traditional RDMBS row (or sub-set of cells in that 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 **asymmetric key pairs** and their identifiers 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; this is enforced cryptographically.

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.