Logarithmic – Application Encryption Documentation

Last updated: November 15, 2025

This document describes how the **Logarithmic** desktop application uses (or does not use) cryptographic functions. It is intended to support export compliance and national encryption control requirements (including distribution in France and via the Apple App Store).

1. Overview of the Application

Logarithmic is a **desktop log viewer** and tailing tool. Its primary purpose is to:

- Read and display log files from the local filesystem.
- Tail log files in real time.
- Optionally stream logs from **Kubernetes** clusters using the user's own kubeconfig.
- Optionally expose log content to local AI tools via a **local-only MCP (Model Context Protocol) HTTP/SSE server**.

The application is not a messaging client, VPN, password manager, disk encryption tool, or any other kind of security / cryptography product.

2. Summary of Encryption Usage

2.1 High-Level Summary

- **Custom cryptography**: **None**. The application does not implement its own encryption algorithms or key management.
- **Data at rest**: **Not encrypted by the application**. Logs and settings are stored in cleartext files on the local filesystem.
- **Network encryption**: Used **only** when connecting to Kubernetes APIs via the standard Kubernetes Python client, which uses HTTPS/TLS under the hood.
- **Local MCP server**: Runs on 127.0.0.1 using plain HTTP + SSE. No TLS is used for this local loopback interface.

2.2 Libraries and APIs Involved

- The application uses the **official kubernetes Python client library** (kubernetes.client, kubernetes.config) to connect to Kubernetes API servers.

- When a kubeconfig file specifies an https:// endpoint, the Kubernetes client uses the system's / Python runtime's standard TLS stack (OpenSSL via urllib3/requests, etc.)
- Logarithmic itself does **not**:
    - Implement or modify any cryptographic algorithms.
    - Ship its own crypto library or native crypto code.
    - Perform any encryption or decryption of user data beyond what the standard HTTPS/TLS stack does internally.

3. Data at Rest

3.1 Log Files

- Logarithmic reads log files from locations explicitly chosen by the user (file picker, configured paths, or wildcards).
- These log files **are not encrypted** by the application.
- The application does **not** create any encrypted containers, volumes, or archives.
- Any protection for these files (e.g., FileVault on macOS) is provided entirely by the operating system, not by Logarithmic.

3.2 Settings and Session Data

- Settings (including session layout, tracked log paths, and UI preferences) are stored in JSON files under the user's home directory (e.g. ~/.logarithmic/).
- These files are **not encrypted** by the application.
- The application does not store authentication secrets, tokens, or passwords.

4. Network Communications

4.1 Kubernetes API Connections

When the user configures a Kubernetes log source, Logarithmic:

1. Prompts the user to select a **kubeconfig** file.
2. Passes the selected kubeconfig path to the Kubernetes Python client (kubernetes.config.load_kube_config).
3. The Kubernetes client then connects to the specified Kubernetes API server.

If the kubeconfig uses an https:// endpoint, all communication with the API server is protected by **standard TLS**:

- TLS protocol versions and ciphersuites are determined by the underlying Python / OpenSSL stack and the remote server.

- Client certificates and keys, if any, are loaded and handled by the Kubernetes client / underlying TLS libraries using the kubeconfig file.
- Logarithmic does **not** implement its own TLS, certificate parsing, or key management; it simply delegates to the upstream library.

Use case:

- The encryption is used solely to protect the confidentiality and integrity of traffic between the user's machine and their chosen Kubernetes API endpoint.
- This use is typical of standard HTTPS-based API clients.

4.2 Local MCP (Model Context Protocol) Server

- Logarithmic optionally runs a local MCP server on 127.0.0.1 (default port 3000).
- This server uses **HTTP with Server-Sent Events (SSE)** to communicate with local AI clients (e.g., Claude Desktop).
- This traffic is restricted to the local loopback interface and does **not** use TLS.
- No encryption is applied by Logarithmic to MCP traffic; access control depends on the local machine's environment and loopback semantics.

4.3 No Other Network Crypto

The application does not:

- Implement VPN tunneling, proxying, or custom encrypted channels.
- Use non-standard or proprietary cryptographic protocols.
- Perform end-to-end content encryption beyond what is provided by standard HTTPS.

5. Keys, Certificates, and Secrets

- Logarithmic **does not generate, store, or manage cryptographic keys** on its own.
- When used with Kubernetes, any client certificates, keys, or cluster CA certificates are referenced in the user's kubeconfig file and are handled by the Kubernetes client library.
- Logarithmic simply passes the kubeconfig path to the library; it does not parse, modify, or copy the keys/certificates.

6. Classification for Export / French Requirements

Based on the above:

1. **Does the application use encryption?**

    - Yes, but only as part of standard HTTPS/TLS connections to Kubernetes API servers via the official Kubernetes Python client, and only when the user configures such a connection.
2. **Does the application implement or contain custom cryptographic algorithms?**
    - No. All cryptographic operations are provided by standard libraries (Python/OpenSSL, Kubernetes client) without modification.
3. **Is encryption used for general-purpose content protection or DRM?**
    - No. Encryption is only used for transport security (standard HTTPS/TLS to Kubernetes APIs). Log files themselves are not encrypted by the application.
4. **Is encryption used for user authentication or channel protection only?**
    - Yes. The TLS usage is for protecting API communication channels (and mutual authentication when the kubeconfig so specifies).
5. **Are there any restrictions or hidden crypto features?**
    - No. All behavior is standard and visible; there are no hidden or undocumented cryptographic capabilities.

7. Changes to This Document

If future versions of Logarithmic introduce additional encryption features (for example, encrypting local configuration or log content), this document will be updated to accurately describe those changes and, if necessary, provide updated information for export control and French distribution requirements.