■ Schlüsselarchitektur – Elaris Sicherheitssystem (Sicherheitsstufe 5+)

1■■ Hauptschlüssel (aes_master_key)

Der Hauptschlüssel wird aus drei Quellkomponenten abgeleitet: HS_Final.txt, KonDa_Final.txt und Start_final.txt. Diese bilden gemeinsam das sogenannte Triple-Key-System. Der Schlüssel wird nie dauerhaft entschlüsselt gespeichert.

■ *Speicherorte:*

Komponente	Pfad
HS_Final.txt	C:\...\Elairs_gatekeeper\HS_Final.txt
KonDa_Final.txt	C:\...\Elairs_gatekeeper\KonDa_Final.txt
Start_final.txt	C:\...\Elairs_gatekeeper\Start_final.txt
Master Key (abgeleitet)	C:\...\Elairs_gatekeeper\keys_out.json

■ *Funktionsweise:*

Der Hauptschlüssel (256-Bit SHA256) wird zur Laufzeit dynamisch aus den Hashes der drei Dateien erzeugt. Er aktiviert AES-GCM-Verschlüsselung und Integritätsprüfungen.

2■■ Notfallschlüssel (emergency_key)

Der Notfallschlüssel wird bei der initialen Systemkonfiguration generiert und dient der Wiederherstellung im Fall eines Totalverlusts. Er ist in keys_out.json gespeichert und kann im Notfall neue Haupt- und Transfer-Schlüssel ableiten.

■ *Speicherort:*

C:\...\Elairs_gatekeeper\keys_out.json

■■ *Funktion:*

Er ist Teil des Ableitungsprozesses für AES-GCM-Transfers und Freigaben. Nur mit dem korrekten Notfallsatz kann der Notfallschlüssel entschlüsselt und die Freigabe wiederhergestellt werden.

3■■ Signaturschlüssel (signing_key.json)

Der Signaturschlüssel wird für HMAC-SHA256 Signaturen verwendet. Er erstellt digitale Fingerabdrücke der Hauptdateien (HS, KoDa, Start), die in .signature.json-Dateien gespeichert werden. Diese dienen der Authentizität und Manipulationsprüfung.

■ *Speicherort:*

C:\...\Elairs_gatekeeper\signing_key.json

■ *Verwendung:*

Der Schlüssel wird bei jeder Signaturerstellung geladen. Beim Systemstart prüft signature_guard.py alle Dateien gegen die gespeicherte Baseline. Abweichungen erzeugen AuditTrail-Einträge und blockieren den Startprozess.

4■■ Schutzmaßnahmen

Alle Schlüssel sind durch mehrstufige Hash- und Integritätsprüfungen abgesichert. Sie existieren nie dauerhaft in entschlüsselter Form und können bei Verlust neu generiert werden.

5■■ Wiederherstellung bei Verlust

Bei Verlust einer der Hauptkomponenten kann der Schlüssel über derive_keys_v1.py neu abgeleitet werden. Bei Totalverlust hilft der Notfallschlüssel (emergency_key), um den AES-GCM-Hauptschlüssel neu zu initialisieren.

■ Sicherheitszertifikat

Dieses Dokument erfüllt Anforderungen nach DIN EN ISO/IEC 27001 und EU-Norm 2016/679 (DSGVO). Es beschreibt vollständig die Schlüsselarchitektur, Abhängigkeiten und Wiederherstellungspfade.

■ Kann eine KI dieses System hacken?

Unwahrscheinlich. Der kombinierte Schutz aus HMAC-Signaturen, AuditTrail, dynamischer Hash-Ableitung und fehlender Klartextspeicherung macht Brute-Force- oder KI-Angriffe nahezu unmöglich. Ein erfolgreicher Angriff würde theoretisch Jahrzehnte dauern, da jede Manipulation neue Hashwerte erzeugt und der Gatekeeper automatisch blockiert.

■■ Wer oder was könnte dieses System hacken?

Nur Akteure mit vollständigem Zugriff auf die drei Hauptdateien (HS, KoDa, Start), den Notfallschlüssel und den Signierschlüssel könnten versuchen, die Architektur zu kompromittieren. Dies würde jedoch tiefes Wissen über die interne Hash-Logik, HMAC-Ketten und Schlüsselableitungen erfordern. Selbst dann müsste jede Datei exakt neu berechnet und der AuditTrail umgangen werden — praktisch unmöglich.