

Kryptokonzept

Verfahren, Schlüssel, Ablage, Wechsel und Laufzeiten

Produktversion 2026.10.03 · Handbuch-Revision 2.1

Stand 03.10.2026 · Plattform .NET 10 / Windows Server

ID-54 Kryptokonzept

Kryptokonzept

Dieses Kapitel beschreibt, **womit** Ordvis IT Daten verschlüsselt, signiert und prüft, **wo** die Schlüssel liegen, **wie** sie gewechselt werden und **wie lange** sie gelten. Es richtet sich an den **IT-Betrieb** und an **Informationssicherheitsbeauftragte**. Es beschreibt den Stand der Fassung, mit der es ausgeliefert wird. Wo das Produkt hinter dem Wünschenswerten zurückbleibt, steht das in Abschnitt 7.

Installation und Härtung stehen in ID-50, Aktualisierungen in ID-50a, Meldewege in ID-53.

INHALT

1 Grundsätze

2 Transportverschlüsselung

3 Ruhende Daten: die Tresore

4 Schlüssel und Token im Einzelnen

4.1 Anmeldung

4.2 Token für Fremdsysteme und Verweise

4.3 Sicherungen

4.4 Aktualisierungen und Lizenzen

4.5 Clients

4.6 Push-Benachrichtigungen

5 Sichern, Wiederherstellen, Wechseln

5.1 Was die Sicherung mitnimmt

5.2 Den Mail-Tresorschlüssel wechseln

6 Wer welchen Schlüssel sehen kann

7 Bekannte Grenzen

8 Beispiele aus der Praxis

8.1 Beispiel: Nach dem Umzug auf neue Hardware sind die Postfach-Kennwörter unlesbar

8.2 Beispiel: Die Schlüssellage für den ISB dokumentieren

9 Häufige Fragen

Verwandte Themen

1 Grundsätze

- **Nur Standardverfahren.** Verschlüsselung mit **AES-256-GCM**, Hashwerte mit **SHA-256**, Signaturen mit **ECDSA P-256 (ES256)**, Schlüsselableitung mit **HKDF-SHA256**. 7-Zip leitet den Schlüssel eines Sicherungsarchivs mit seinem eigenen SHA-256-Verfahren aus dem Kennwort ab.
- **Keine eigene Kryptografie.** Die Verfahren stammen aus der .NET-Laufzeit und aus geprüften Bibliotheken: Argon2id aus *Isopoh.Cryptography.Argon2*, der Zertifikatsbezug (ACME) aus *Certes*, SNMPv3 aus *SharpSnmpLib*, die Prüfung von OpenPGP-Signaturen aus *BouncyCastle*; dazu 7-Zip für die Sicherungen.
- **Zufall nur aus dem Betriebssystem.** Schlüssel, Token und die Kennwörter, die das Setup erzeugt, stammen aus dem kryptografischen Zufallsgenerator von Windows.
- **Geheimnisse liegen nicht lesbar in der Datenbank.** Zugangsdaten sind verschlüsselt. Token, die nur geprüft werden müssen, liegen ausschließlich als SHA-256-Hashwert vor. Ausnahmen nennt Abschnitt 7.
- **Schlüssel auf dem Server** sind, wo das Produkt sie selbst anlegt, mit **Windows-DPAPI** geschützt – an das Konto des Dienstes oder an den Rechner gebunden. Ausnahmen nennt Abschnitt 7.

2 Transportverschlüsselung

Verbindung	Verfahren	Prüfung der Gegenseite
Client, Wandschirm, App → Schnittstelle (API)	TLS 1.2 und 1.3 (einstellbar über <code>Security:Tls:Protocols</code>); die Cipher-Suites legt Windows (Schannel) bzw. die Gruppenrichtlinie fest	Zertifikatskette des Systems oder ein hinterlegter SHA-256 -Fingerabdruck (Vergleich in fester Laufzeit)
Browser → Portal	TLS nach der Vorgabe von Windows (Schannel, Gruppenrichtlinie); <code>Security:Tls:Protocols</code> wirkt hier nicht	durch den Browser
Portal → API	TLS	Systemkette oder hinterlegter SHA-256-Fingerabdruck
Server → Verzeichnisdienst (LDAPS)	TLS	mit Org-CA: Kette zu ihr; ohne Org-CA: Kette zum Windows-Zertifikatsspeicher und eingetragener Hostname (ID-41). Eine Sperrprüfung findet nicht statt
Server → Mail (IMAP, POP3, SMTP), Exchange (EWS)	TLS	Standard: Systemkette. „Eigene CA“: Fingerabdruck SHA-256 (empfohlen) oder SHA-1; ein Namensfehler wird nie angenommen
Server → Push-Vermittler → Google	TLS; der Inhalt ist zusätzlich Ende-zu-Ende verschlüsselt (Abschnitt 4.6)	Systemkette
Server → PostgreSQL	Npgsql-Standard; das Produkt erzwingt keine Verschlüsselung (Abschnitt 7)	—

Achtung – Wege ohne Verschlüsselung, die Sie selbst einschalten können: Ein Verzeichnisdienst ohne SSL bindet mit Kennwort im Klartext (Port 389). Bei Postfächern und Exchange nimmt die Option „Ungültige akzeptieren“ jedes Zertifikat an, die Sicherheit „Keine“ verbindet unverschlüsselt. Webhook-Ziele dürfen `http:` sein. SNMPv2c überträgt die Community im Klartext. Nutzen Sie diese Wege nur, wo das Netz selbst geschützt ist.

Serverzertifikat. Drei Betriebsarten (ID-50, Abschnitt 7.1):

Betriebsart	Schlüssel	Ablage	Wechsel / Laufzeit
Eigenes Zertifikat (PFX oder PEM)	wie ausgestellt	PFX-Datei; ihr Kennwort DPAPI-geschützt in <code>hosting.json</code> . Ein eingefügtes PEM wird als <code>%ProgramData%\InfraDesk\config\certs\server.pfx</code> mit Zufallskennwort abgelegt, lesbar nur für SYSTEM und Administratoren	durch den Betreiber
Windows-Zertifikatsspeicher	wie ausgestellt	<code>LocalMachine\My</code> , gefunden über den Fingerabdruck	durch den Betreiber

Betriebsart	Schlüssel	Ablage	Wechsel / Laufzeit
Let's Encrypt (ACME, HTTP-01)	ECDSA P-256	%ProgramData%\InfraDesk\acme\certificate.pfx.dpapi : Zertifikat und zufälliges Kennwort in einer DPAPI-geschützten Datei	automatisch, sobald die Restlaufzeit unter 30 Tagen liegt; geprüft alle 12 Stunden

Hinter einem **Reverse-Proxy** übernimmt dieser TLS; der Server selbst spricht dann HTTP.

3 Ruhende Daten: die Tresore

Ordvis IT hat zwei Tresorschlüssel. Beide verschlüsseln mit **AES-256-GCM**: 96-Bit-Nonce aus dem Zufallsgenerator je Wert und 128-Bit-Prüfsumme (Tag). Ein veränderter Wert lässt sich nicht mehr entschlüsseln.

	Tresorschlüssel	Mail-Tresorschlüssel
Schützt	LDAP-Bind-Kennwort, OIDC-Client-Geheimnis, SSO-Sitzungen, Webhook-Geheimnisse der Alarmierung, Scan-Zugangsdaten der Discovery (auch SNMPv3), TOTP-Geheimnisse, den NVD-Zugangsschlüssel, Exchange-Verbindungen der Selbstorganisation	Postfach-Kennwörter und OAuth-Token, Messenger-Konten, Action1-Verbindungen
Schlüssel	256 Bit	256 Bit
Quelle	CredentialVault:EncryptionKey (Base64) oder die Datei %ProgramData%\InfraDesk\secret.key	MailVault:EncryptionKey , sonst CredentialVault:EncryptionKey , sonst die Datei <Programm>\api\data\mail-vault.key
Ablage der Datei	DPAPI-geschützt (Konto des Dienstes); eine Klartextdatei stellt der Dienst beim Start um	Base64-Text (Abschnitt 7)
Wechsel	kein Mechanismus. Ein neuer Schlüssel macht alles Gespeicherte unlesbar	im Client, nur wenn der Schlüssel aus der Datei stammt (Abschnitt 5.2)
Laufzeit	unbegrenzt	bis zum Wechsel

Konfigurierte Schlüssel: Beide Einträge (und Ticketing:ReplyTokenSecret) dürfen DPAPI-geschützt stehen, Präfix **DPAPI:** , geschützt unter dem Dienstkonto. Ein Klartextwert wird bei jedem Start als Warnung gemeldet; ein geschützter, aber unlesbarer Wert lässt den Tresor benannt scheitern, statt still auf die Datei auszuweichen (ID-50, Abschnitt 3.7).

Achtung: Ist CredentialVault:EncryptionKey gesetzt und MailVault:EncryptionKey nicht, gilt ein Schlüssel für beide Tresore – und zugleich für die Vorgangsmarke in Antwortmails (Abschnitt 4.2). Ihn nachträglich zu setzen oder zu ändern macht dann auch die Postfach-Zugangsdaten unlesbar und ordnet Antworten auf ältere Mails keinem Vorgang mehr zu.

Achtung – alter Schlüsselname: Die Discovery liest zuerst den älteren Eintrag `CredentialVault:EncryptionKeyBase64`, der übrige Tresor kennt ihn nicht. Steht er noch in Ihrer Konfiguration, haben Scan-Zugangsdaten und die übrigen Geheimnisse verschiedene Schlüssel. Stellen Sie auf `CredentialVault:EncryptionKey` um.

Wichtig: Beide Schlüssel gehören in die Sicherung (Abschnitt 5). Wer sie verliert, muss alle genannten Zugangsdaten neu eingeben und die MFA neu einrichten lassen.

4 Schlüssel und Token im Einzelnen

4.1 Anmeldung

Gegenstand	Verfahren	Erzeugung und Ablage	Wechsel / Laufzeit
Kennwörter lokaler Konten	Argon2id , 64 MiB, 3 Durchläufe, Parallelität 1	Hashwert mit Salt in der Datenbank	bei Kennwortänderung
Zugriffstoken (JWT)	HMAC-SHA256 (HS256)	Signaturgeheimnis: 48 Zeichen aus 64 (288 Bit) vom Setup; verschlüsselt in der Registry unter <code>HKLM\SOFTWARE\Ordavis\IT\Geheimnisse</code> (DPAPI des Rechners, lesbar nur für SYSTEM und Administratoren), übergangsweise zusätzlich als Maschinen-Umgebungsvariable <code>Identity__Jwt__Secret</code> . Ein erneuter Setup-Lauf behält ein vorhandenes Geheimnis mit mindestens 32 Zeichen	Token: 15 Minuten (App: 10 Minuten). Geheimnis: kein Wechselmechanismus (Abschnitt 7)
Erneuerungstoken	64 Byte Zufall	nur der SHA-256 -Hashwert wird gespeichert; jede Erneuerung ersetzt das Token	12 Stunden ohne Nutzung (gleitend), ohne feste Höchstdauer
Zweiter Faktor (TOTP)	RFC 6238: HMAC-SHA1, 6 Stellen, 30 Sekunden	160-Bit-Geheimnis, verschlüsselt mit dem Tresorschlüssel	bis zur Neueinrichtung
Geräteanmeldung der App	ECDSA P-256	privater Schlüssel im Android-Keystore (StrongBox, wo vorhanden); er verlässt das Gerät nicht	bis zum Entfernen des Geräts
Windows-SSO (Vertrauensgeheimnis zwischen Portal und Server)	Vergleich in fester Laufzeit	in der Konfiguration, vorgesehen DPAPI-geschützt (Konto des Dienstes); ein Klartextwert wird angenommen	durch den Betreiber

Gegenstand	Verfahren	Erzeugung und Ablage	Wechsel / Laufzeit
OIDC	Token des Anbieters, geprüft über dessen JWKS: Aussteller, Zielgruppe, Laufzeit, <code>nonce</code> ; PKCE (S256) ist Pflicht. Vorgabe RS256/384/512 und ES256/384; HS* und <code>none</code> sind ausgeschlossen	Client-Geheimnis im Tresor	beim Anbieter
Portal-Sitzung	Anmelde-Cookie des Portals	Schlüsselring nach Vorgabe von ASP.NET Core (Abschnitt 7)	8 Stunden gleitend (<code>Session:LifetimeMinutes</code>)

4.2 Token für Fremdsysteme und Verweise

Gegenstand	Verfahren	Ablage
Integrations-Token (API-Schlüssel), Kalender-Abonnements, Wandschirm-Kopplung, Kennwort-Zurücksetzen, Collector-Anmeldung	32 Byte Zufall	nur der SHA-256 -Hashwert
Webhook-Signatur der Alarmierung	HMAC-SHA256 über Zeitstempel und Inhalt	Geheimnis im Tresor, zusätzlich sein Hashwert
Vorgangsmarke in Antwortmails	HMAC-SHA256 , gekürzt auf 48 Bit	Schlüssel <code>Ticketing:ReplyTokenSecret</code> , sonst <code>CredentialVault:EncryptionKey</code> , sonst Datei <code><Programm>\api\data\reply-token.key</code>
Prüfwert auf Inventaretiketten	HMAC-SHA256 , gekürzt auf 50 Bit; Schlüssel per HKDF-SHA256 aus dem JWT-Geheimnis abgeleitet	nicht gespeichert, jederzeit neu berechnet

Ein Wandschirm-Token gilt 365 Tage (höchstens 730) und gibt dem Wandschirm die Rechte seines Anzeigekontos ohne Kennwort und ohne zweiten Faktor (ID-27b).

4.3 Sicherungen

Gegenstand	Verfahren	Schlüssel
Datenbanksicherung (<code>backup.ps1</code>)	7-Zip, AES-256, Dateinamen mitverschlüsselt	Kennwort <code>Backup:Password</code> in <code>operations.json</code> , DPAPI-geschützt. Ohne Kennwort bleibt die Sicherung unverschlüsselt

Gegenstand	Verfahren	Schlüssel
Beipack der Sicherung	im selben Archiv – nur wenn ein Kennwort gesetzt ist	enthält die Schlüsseldateien entschlüsselt , damit eine Wiederherstellung auf anderer Hardware gelingt (Abschnitt 5.1)
BCM-Offline-Notfall-Kit	7-Zip, AES-256 , Dateinamen mitverschlüsselt	Passphrase des Betreibers, im Client gesetzt und DPAPI-geschützt in <code>operations.json</code> (Rückfall: Umgebungsvariable <code>Bcm__OfflineKit__Passphrase</code>); ohne sie entsteht kein Kit (ID-29)

4.4 Aktualisierungen und Lizenzen

Gegenstand	Verfahren	Schlüssel
Aktualisierungsmanifest	ECDSA P-256 (JWS ES256); jede Datei zusätzlich über ihren SHA-256 -Wert geprüft	öffentlicher Schlüssel fest im Programm (Serverdienst, Updater, Aktualisierungsdienste, App); der private beim Hersteller (kein HSM)
Installationspakete	keine Authenticode-Signatur; SHA-256 -Prüfsummen im Download-Bereich (ID-50, Abschnitt 2.2.1)	—
Lizenzen	ECDSA P-256 (ES256)	öffentliche Schlüssel fest im Programm; der private beim Hersteller (kein HSM)
Aktivierung, Nutzungsfrist und Testzeitraum	ECDSA P-256 (ES256)	signiert vom Aktivierungsdienst des Herstellers

4.5 Clients

Gegenstand	Ablage
Windows-Client	Gespeichert wird nur das Erneuerungstoken einer SSO-Anmeldung , je Server, mit DPAPI des angemeldeten Benutzers; eine Kennwortanmeldung wird nicht gespeichert
Wandschirm	Token mit DPAPI des angemeldeten Benutzers
Collector (Discovery)	Zugang mit DPAPI des Rechners – jeder Prozess auf diesem Rechner, der die Datei lesen darf, kann ihn entschlüsseln
App	gesicherter Speicher von Android (SecureStorage)

4.6 Push-Benachrichtigungen

Betreff und Text einer Benachrichtigung sind **Ende-zu-Ende verschlüsselt**, wenn die App einen Schlüssel gemeldet hat: **ECDH P-256** mit einem Einmalschlüssel je Nachricht, **HKDF-SHA256**, **AES-256-GCM**. Google und der Push-Vermittler sehen nur einen Platzhalter („Neue Meldung“, „Alarm“). Der private Schlüssel liegt im gesicherten Speicher der App. Der Vermittler speichert Gerätetoken nur als SHA-256-Hashwert.

Hinweis: Unverschlüsselt gehen mit: der Verweis in die App (Deep-Link), der Ereignistyp und die Dringlichkeit. Ein Gerät, das noch keinen Schlüssel gemeldet hat, bekommt Betreff und Text im Klartext.

5 Sichern, Wiederherstellen, Wechseln

5.1 Was die Sicherung mitnimmt

Diese Schlüssel müssen eine Wiederherstellung überleben. Das verschlüsselte Beipack der Sicherung nimmt sie mit:

Schlüssel	Ohne ihn
Tresorschlüssel (<code>secret.key</code>)	Verzeichnis-, SSO-, Webhook-, Scan-, NVD- und Exchange-Zugangsdaten unlesbar; MFA muss neu eingerichtet werden
Mail-Tresorschlüssel (<code>mail-vault.key</code>)	Postfach-, Messenger- und Action1-Zugangsdaten unlesbar
Vorgangsmarke (<code>reply-token.key</code>)	Antworten auf ältere Mails werden keinem Vorgang mehr zugeordnet

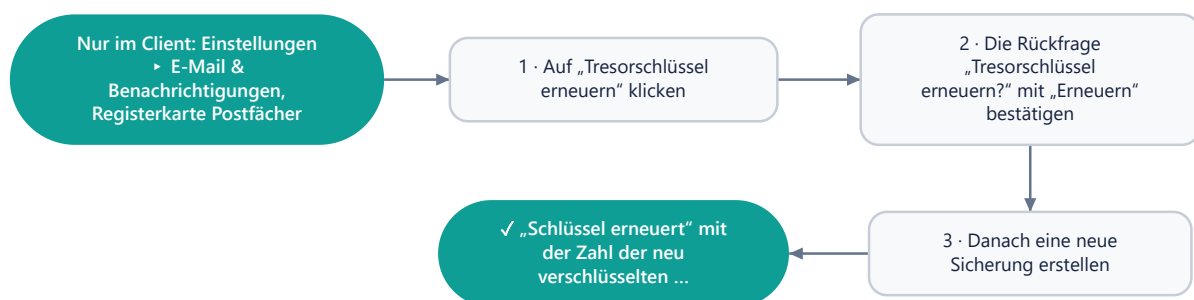
Achtung – drei Bedingungen: Das Beipack entsteht nur, wenn ein **Sicherungskennwort** gesetzt ist. Es enthält nur Schlüssel aus **Dateien** – Schlüssel aus der Konfiguration sichern Sie selbst. Und die Dateien `mail-vault.key` und `reply-token.key` findet die Sicherung nur, wenn das Programm unter `C:\` oder `D:\Program Files\Ordivis Platform` (bzw. dem älteren `...\InfraDesk`) liegt. Bei einer Installation auf einem anderen Laufwerk sichern Sie `<Programm>\api\data\*.key` selbst.

Das **JWT-Geheimnis** liegt in der Registry (`HKLM\SOFTWARE\Ordivis\IT\Geheimnisse`) und übergangsweise in der Umgebung des Servers, nicht im Beipack. Geht es verloren, genügt eine neue Anmeldung aller Benutzer; die Prüferte **gedruckter Etiketten** werden dann jedoch ungültig.

5.2 Den Mail-Tresorschlüssel wechseln

 **Nur im Client:** *Einstellungen* ▶ *E-Mail & Benachrichtigungen*, Registerkarte *Postfächer* · 
`notifications.mailbox.admin` und `system.config.manage` — Das Portal bietet diesen Wechsel nicht.

Voraussetzung: Der Schlüssel stammt aus der Datei `mail-vault.key`, nicht aus der Konfiguration.



Klickweg – Den Mail-Tresorschlüssel wechseln

1. Klicken Sie auf *Tresorschlüssel erneuern*.
2. Bestätigen Sie die Rückfrage „Tresorschlüssel erneuern?“ mit *Erneuern*.
3. Erstellen Sie danach eine neue Sicherung (ID-50, Kapitel 6).

✓ **Ergebnis:** „Schlüssel erneuert“ mit der Zahl der neu verschlüsselten Geheimnisse. Alle Geheimnisse aller Mandanten sind in **einer** Transaktion neu verschlüsselt; der alte Schlüssel bleibt als `mail-vault.key.previous` liegen. Das Audit-Log hält den Wechsel fest (`mail.vault.key.rotated`).

Lässt sich ein Geheimnis nicht entschlüsseln, bricht der Wechsel ab: „Mindestens ein Geheimnis liess sich nicht entschlüsseln — Rotation abgebrochen, nichts geändert.“ Stammt der Schlüssel aus der Konfiguration, lehnt der Server ab: „Der Vault-Schlüssel stammt aus der Konfiguration (MailVault:EncryptionKey) und wird dort rotiert.“

Weitere Wechsel: Das Serverzertifikat wechselt bei ACME von selbst, sonst durch den Betreiber. Für Tresorschlüssel, JWT-Geheimnis und Vorgangsmarke gibt es keinen Wechselmechanismus.

6 Wer welchen Schlüssel sehen kann

Wer **Verwaltungsrechte auf dem Server** hat, kann jeden Schlüssel dieses Kapitels erreichen: DPAPI-geschützte Dateien über das Konto des Dienstes, die Registry-Geheimnisse über die Zugriffsliste, Umgebungsvariablen direkt. Der Schutz richtet sich gegen Leser einer Datenbanksicherung und gegen jemanden, der eine einzelne DPAPI-geschützte Datei vom Server kopiert.

Schlüsseldateien und lokale Benutzer. Das Setup gibt der Gruppe *Benutzer* Leserecht auf das Programmverzeichnis – sonst startet der Client nicht –, den Ordner `api\data` nimmt es davon aus. `mail-vault.key`, `mail-vault.key.previous`, `reply-token.key` und der ACME-Kontoschlüssel `account.pem` tragen eine eigene Zugriffsliste nur für SYSTEM und Administratoren. Eine Datei, die vor dem Update vom Oktober 2026 entstanden ist, schützt der Dienst, sobald er sie das nächste Mal lädt; gelingt das nicht, steht eine Warnung im Protokoll des Dienstes.

Achtung – die Maschinen-Umgebungsvariablen (Abschnitt 7) kann weiterhin **jedes lokale Konto** auf dem Server lesen. Lassen Sie auf dem Ordavis-Server nur Administratoren interaktiv zu.

7 Bekannte Grenzen

Offen gesagt, damit Sie sie in Ihrer Risikobewertung berücksichtigen können:

- **JWT-Geheimnis und Datenbankkennwort** liegen verschlüsselt in der Registry (`HKLM\SOFTWARE\Ordavis\IT\Geheimnisse`, DPAPI, lesbar nur für SYSTEM und Administratoren). Der Dienst übernimmt sie beim Start selbst dorthin. **Zusätzlich** stehen sie im Klartext in den Maschinen-Umgebungsvariablen `Identity__Jwt__Secret` und `ConnectionStrings__InfraDesk`, die **jedes lokale Konto** lesen kann. Jeder Setup-Lauf schreibt sie dort neu; die Fassung dieses Handbuchs entfernt sie nicht von selbst.
- **Die Zugangsübersicht** `%ProgramData%\InfraDesk\setup-credentials.txt` (nur für Administratoren lesbar) enthält Datenbank- und PostgreSQL-Kennwörter im Klartext. Löschen Sie diese Datei, nachdem Sie die Angaben sicher verwahrt haben. Bis dahin zeigt der Systemstatus die Zeile „Zugangsdatei der Einrichtung“ als gestört; der Gesamtstatus bleibt betriebsbereit.
- **Kein Wechselmechanismus** für Tresorschlüssel, JWT-Geheimnis und Vorgangsmarke.
- `mail-vault.key` und `reply-token.key` liegen als Base64-Text im Programmverzeichnis, lesbar nur für SYSTEM und Administratoren (Abschnitt 6). Ist eine der Dateien unlesbar oder hat sie die falsche Länge, legt der Dienst eine neue an, überschreibt die alte und meldet nur eine Warnung im Protokoll – die damit verschlüsselten Geheimnisse sind dann verloren.
- **Weitere Geheimnisse außerhalb der Tresore:** Das OIDC-Client-Geheimnis des Portals (`oidc:ClientSecret`) und das Kennwort des Rückfall-Mailserver (`Notifications:Smtp:Password`) stehen im Klartext in der Konfiguration.

Der Zugangsschlüssel für den NetBox-Import liegt DPAPI-geschützt in der Datenbank und lässt sich auf anderer Hardware nicht wiederherstellen.

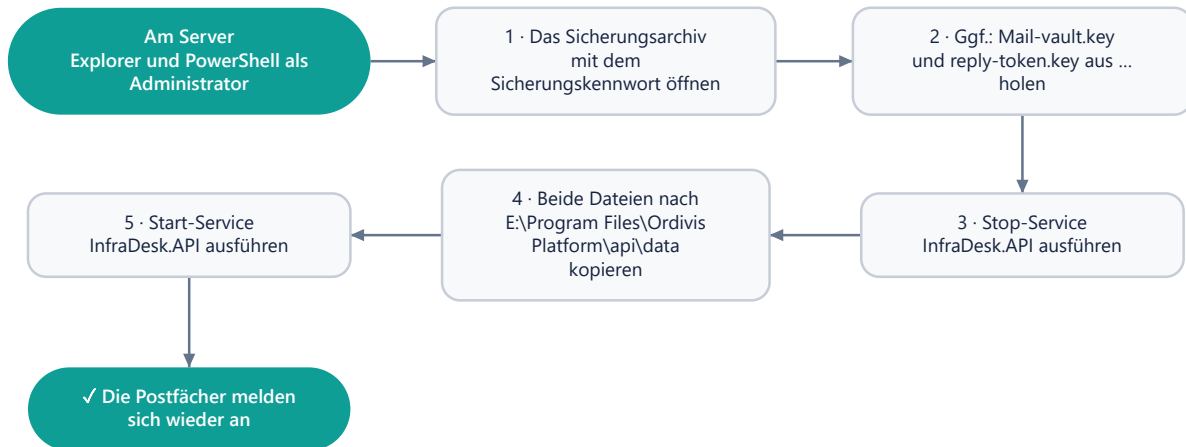
- Der **ACME-Kontoschlüssel** (`account.pem`) liegt im Klartext neben dem Zertifikat, lesbar nur für SYSTEM und Administratoren.
- Die Verbindung zu **PostgreSQL** wird nicht erzwungen verschlüsselt. Betreiben Sie die Datenbank auf demselben Rechner oder erzwingen Sie TLS in PostgreSQL selbst (`pg_hba.conf` : `hostssl`).
- Das **Sicherungskennwort** und die Passphrase des BCM-Notfall-Kits stehen während des Packens in der Kommandozeile des 7-Zip-Prozesses. Sichtbar ist das dort nur für Konten mit Verwaltungsrechten.
- **Tresorwerte sind nicht an ihre Zeile gebunden.** Wer Schreibzugriff auf die Datenbank hat, kann ein verschlüsseltes Geheimnis in eine andere Zeile kopieren, ohne dass es auffällt. Ein unlesbarer Wert im Tresor der Identitätsverwaltung gilt als leer.
- **SAML** ist nicht freigegeben und abgeschaltet: Jede Adresse unter `/sso/saml` antwortet mit 410 und verarbeitet nichts. Nutzen Sie OIDC oder Windows-SSO.
- Der **Collector** prüft das Serverzertifikat gegen den Zertifikatsspeicher oder gegen einen im Setup hinterlegten **SHA-256-Fingerabdruck**. Collector, die früher „Zertifikatsprüfung abschalten“ trugen, merken sich beim ersten Kontakt nach dem Update das Zertifikat, das sie sehen („trust on first use“): Steht genau in diesem Moment jemand dazwischen, wird dessen Zertifikat gemerkt. Gleichen Sie den gemerkten Fingerabdruck deshalb einmal mit dem Serverzertifikat ab (ID-24a).
- **TOTP** nutzt HMAC-SHA1, weil es die Authenticator-Apps so verlangen (RFC 6238).
- Der **Schlüsselring des Portals** (Anmelde-Cookies) folgt den Vorgaben von ASP.NET Core; das Produkt legt Ort und Schutz nicht selbst fest.
- Die **privaten Signierschlüssel des Herstellers** (Manifeste, Lizenzen, Aktivierung) liegen nicht in einem Hardware-Sicherheitsmodul.

8 Beispiele aus der Praxis

8.1 Beispiel: Nach dem Umzug auf neue Hardware sind die Postfach-Kennwörter unlesbar

Ausgangslage: Die Stadtverwaltung Musterstadt hat den Ordavis-Server auf neue Hardware umgezogen und dort eine Sicherung zurückgespielt. Das Programm liegt jetzt unter `E:\Program Files\Ordavis Platform`. Die Postfächer melden, dass ihre Kennwörter nicht entschlüsselt werden können.

 **Am Server:** Explorer und PowerShell als Administrator ·  lokaler Administrator



Klickweg – Beispiel: Nach dem Umzug auf neue Hardware sind die Postfach-Kennwörter unlesbar

1. Öffnen Sie das Sicherungsarchiv mit dem Sicherungskennwort und prüfen Sie, ob es den Ordner `api-data` mit `mail-vault.key` enthält.
2. Fehlt er, holen Sie `mail-vault.key` und `reply-token.key` aus `<Programm>\api\data` des alten Servers.
3. Führen Sie `Stop-Service InfraDesk.API` aus.
4. Kopieren Sie beide Dateien nach `E:\Program Files\Ordivis Platform\api\data`.
5. Führen Sie `Start-Service InfraDesk.API` aus.

✓ **Ergebnis:** Die Postfächer melden sich wieder an. Sichern Sie `api\data\*.key` künftig selbst mit, denn auf Laufwerk E: findet die Sicherung diese Dateien nicht (Abschnitt 5.1).

8.2 Beispiel: Die Schlüssellage für den ISB dokumentieren

Ausgangslage: Der Informationssicherheitsbeauftragte der Beispiel GmbH fragt, welche Schlüssel es gibt, wer sie lesen kann und was beim Verlust geschieht.

🔧 **Am Server:** PowerShell als Administrator · 🔑 lokaler Administrator

1. Prüfen Sie mit `Test-Path "$env:ProgramData\InfraDesk\secret.key"`, ob der Tresorschlüssel aus der Datei stammt.
2. Prüfen Sie in `<Programm>\api\appsettings.json`, ob `CredentialVault:EncryptionKey`, `MailVault:EncryptionKey` oder `Ticketing:ReplyTokenSecret` gesetzt sind und mit `DPAPI:` beginnen.
3. Prüfen Sie, ob `%ProgramData%\InfraDesk\setup-credentials.txt` noch vorhanden ist.
4. Übernehmen Sie die Tabellen der Abschnitte 3, 5.1 und 7 in Ihre Risikobewertung.

✓ **Ergebnis:** Der ISB hat eine vollständige Übersicht über Schlüssel, Ablage, Leserkreis und Folgen eines Verlusts. Klartextwerte in der Konfiguration und eine noch vorhandene Zugangsdatei sind als Maßnahmen vermerkt.

9 Häufige Fragen

Welche Verfahren setzt Ordivis IT ein? AES-256-GCM, SHA-256, ECDSA P-256, HMAC-SHA256, HKDF-SHA256 und Argon2id; für Sicherungen 7-Zip mit AES-256 (Abschnitt 1).

Liegen Kennwörter im Klartext in der Datenbank? Nein. Zugangsdaten sind verschlüsselt, Kennwörter lokaler Konten als Argon2id-Hash gespeichert, Token nur als SHA-256-Hash.

Kann ich den Tresorschlüssel wechseln? Nur den Mail-Tresorschlüssel, und nur wenn er aus der Datei stammt (Abschnitt 5.2). Für den übrigen Tresor gibt es keinen Wechselmechanismus.

Was passiert, wenn `secret.key` verloren geht? Alle damit verschlüsselten Zugangsdaten sind unlesbar; die MFA muss neu eingerichtet werden. Deshalb gehört er ins Beipack der Sicherung (Abschnitt 5.1).

Ist die Sicherung verschlüsselt? Nur wenn ein Sicherungskennwort gesetzt ist. Ohne Kennwort entsteht auch kein Beipack mit den Schlüsseln.

Ist die Verbindung zur Datenbank verschlüsselt? Nicht erzwungen. Betreiben Sie PostgreSQL auf demselben Rechner oder erzwingen Sie TLS in PostgreSQL (Abschnitt 7).

Welche TLS-Fassungen gelten? An der Schnittstelle TLS 1.2 und 1.3 (`Security:TLs:Protocols`), am Portal die Vorgabe von Windows (Abschnitt 2).

Sieht Google den Inhalt von Push-Nachrichten? Nein, wenn die App ihren Schlüssel gemeldet hat; Google sieht dann nur einen Platzhalter (Abschnitt 4.6).

Sind die Installer signiert? Nicht mit Authenticode. Prüfen Sie die SHA-256-Prüfsumme (ID-50, Abschnitt 2.2.1). Die Updates selbst sind mit ES256 signiert.

Wer kann die Schlüssel auf dem Server lesen? Administratoren alle. Die Schlüsseldateien (`mail-vault.key` , `reply-token.key` , `account.pem`) nur SYSTEM und Administratoren; die Maschinen-Umgebungsvariablen jedes lokale Konto (Abschnitt 6).

Verwandte Themen

- ID-50 – Betriebshandbuch – Installation, Härtung, Sicherung, Geheimnisse und Schlüssel
- ID-50a – Update-System – Signatur und Prüfsummen der Aktualisierungen
- ID-53 – Produktsicherheit & Meldewege – Meldewege und Außerbetriebnahme
- ID-55 – Technische und organisatorische Maßnahmen – Einordnung der Verschlüsselung
- ID-41 – Identität, Benutzer & Rollen – LDAPS, MFA, SSO
- ID-29 – BCM – Offline-Notfall-Kit