Data Encryption Policy

Data Encryption Policy

> **Project / Organization:** Bilko — Balkan Accounting SaaS **Policy Number:** POL-SEC-ENC-001 **Version:** 1.0 **Date:** 2026-02-23 **Author:** CTO / Security Architect **Status:** Draft **Reviewers:** DPO, Engineering Lead **Classification:** Confidential

Document History

Version	Date	Author	Changes
0.1	2026-02-23	CTO	Initial encryption policy for Bilko accounting SaaS

---

1. Purpose & Scope

This policy defines encryption standards for all data processed by Bilko. Bilko handles regulated financial data across three jurisdictions (Serbia, Bosnia & Herzegovina, Croatia) including tax IDs, IBAN numbers, and financial transaction records that must meet GDPR, ZZPL, and ZZLP BiH requirements.

Scope: All Bilko systems, databases, APIs, backups, and data in transit.

---

2. Data Classification & Encryption Requirements

Level	Label	Examples	Encryption Required

L4-A	Restricted (Personal)	JMBG, OIB	**AES-256-GCM field-level encryption** (prisma-field-encryption) + HMAC-SHA256 hash column + AES-256 at-rest + TLS 1.3 (See ADR-014)
L4-B	Restricted (Business/Financial)	PIB, JIB, IBAN	**AES-256 disk-level encryption** (Railway) + TLS 1.3 + org-scoping + RBAC + API masking for IBAN (last 4 digits in list views) (See ADR-014)
L3	Confidential	Invoice amounts, bank statements, transaction data	**AES-256 at-rest + TLS 1.3**
L2	Internal	Email, name, phone, address	**TLS 1.3** minimum
L1	Public	Organization display name, public invoice ref	No encryption required (but TLS in transit)

---

3. Encryption-in-Transit

Standards

- **Minimum:** TLS 1.2 (legacy client support only)
- **Preferred:** TLS 1.3 — enforced via Cloudflare SSL Full (Strict) mode
- **Cipher suites (TLS 1.3):** TLS_AES_256_GCM_SHA384, TLS_CHACHA20_POLY1305_SHA256
- **Certificate:** Let's Encrypt via Cloudflare (auto-renew) for *.bilko.io
- **HSTS:** `Strict-Transport-Security: max-age=31536000; includeSubDomains; preload`

Scope

Connection	Encryption
Browser → Cloudflare	TLS 1.3 (Cloudflare managed)
Cloudflare → Railway API	TLS 1.2+ (Full Strict mode)
API → PostgreSQL (Railway)	`ssl=require` in DATABASE_URL connection string
API → SEF portal (Serbia)	TLS 1.2+ (Serbian government portal)
API → FINA/HR-FISK (Croatia)	TLS 1.2+ (FINA PKI)

Connection	Encryption
API → Sentry	TLS 1.3

What is NOT acceptable

- HTTP (unencrypted) for any Bilko endpoint — Cloudflare redirects HTTP → HTTPS
- Self-signed certificates in production
- TLS 1.0 or TLS 1.1 connections
- Disabling certificate verification in API clients

---

4. Encryption-at-Rest

Database (PostgreSQL on Railway)

- **Algorithm:** AES-256 (Railway managed disk encryption)
- **Level:** Full disk encryption — Railway EU West
- **Backups:** Encrypted using same key before upload to Railway backup storage
- **Connection:** `ssl=require` — plaintext connection not allowed even from same host

Backup Files

- Bilko does not manage its own database backups — Railway handles this
- If manual exports are taken for disaster recovery: encrypt with GPG (AES-256) before storing
- GPG key stored in Vaultwarden, not in same location as backup files

---

5. Field-Level Encryption (L4-A: JMBG and OIB Only)

> **Tiered L4 approach per ADR-014:** Field-level encryption applies ONLY to L4-A personal identifiers (JMBG, OIB). L4-B fields (PIB, JIB, IBAN) rely on disk-level encryption and application controls — see Section 5b. Field-level encryption for publicly available business tax IDs (PIB, JIB) is disproportionate per GDPR Article 32.

Field-level encryption is applied to JMBG and OIB BEFORE writing to the database. The database column stores only ciphertext. The application decrypts on read.

Algorithm

- **Algorithm:** AES-256-GCM (authenticated encryption — provides confidentiality + integrity)
- **IV:** 12 bytes, randomly generated per encryption operation
- **Key:** 32 bytes (256 bits), stored in `FIELD_ENCRYPTION_KEY` environment variable (Railway secret)
- **Output format:** `base64(iv):base64(authTag):base64(ciphertext)` stored as TEXT column

Implementation

```
import { createCipheriv, createDecipheriv, randomBytes } from "crypto";

const ALGORITHM = "aes-256-gcm";

function getKey(): Buffer {
  const hex = process.env.FIELD_ENCRYPTION_KEY;
  if (!hex || hex.length !== 64) {
    throw new Error("FIELD_ENCRYPTION_KEY must be a 64-character hex string (32 bytes)");
  }
  return Buffer.from(hex, "hex");
}

export function encryptField(plaintext: string): string {
  const key = getKey();
  const iv = randomBytes(12);
  const cipher = createCipheriv(ALGORITHM, key, iv);
  const encrypted = Buffer.concat([cipher.update(plaintext, "utf8"), cipher.final()]);
  const authTag = cipher.getAuthTag();
  return [iv, authTag, encrypted].map(b => b.toString("base64")).join(":");
}

export function decryptField(ciphertext: string): string {
  const key = getKey();
  const [ivB64, tagB64, encB64] = ciphertext.split(":");
  const iv = Buffer.from(ivB64, "base64");
  const authTag = Buffer.from(tagB64, "base64");
```

```
  const encrypted = Buffer.from(encB64, "base64");
  const decipher = createDecipheriv(ALGORITHM, key, iv);
  decipher.setAuthTag(authTag);
  return Buffer.concat([decipher.update(encrypted), decipher.final()]).toString("utf8");
}
```

Fields Subject to Field-Level Encryption (L4-A)

Field	Table	Jurisdiction	Notes
`jmbg` (JMBG)	Contact	RS, BA	Serbian/BiH citizen number — irrevocable personal identifier, encodes DOB/gender/region. Stored encrypted + `jmbgHash` HMAC column.
`oib` (OIB)	Contact	HR	Croatian personal/company tax ID — unique cross-system identifier. Stored encrypted + `oibHash` HMAC column.

Fields NOT Subject to Field-Level Encryption (L4-B — Disk-Level + Controls)

Per ADR-014, the following L4 fields are protected by disk-level encryption (Railway AES-256) plus application-layer controls rather than field-level encryption. Field-level encryption for these fields is disproportionate: PIB and JIB are publicly searchable on government registries; IBAN is routinely shared for payment.

Field	Table	Jurisdiction	Controls
`taxId` / `registrationNumber` (PIB)	Contact, Organization	RS	Disk encryption + org-scoping + RBAC + TLS
`taxId` / `registrationNumber` (JIB)	Contact, Organization	BA	Disk encryption + org-scoping + RBAC + TLS
`iban`	BankAccount	All	Disk encryption + org-scoping + RBAC + TLS + API masking (last 4 digits in list views)

Searchability

L4-A encrypted fields (JMBG, OIB) cannot be searched with SQL LIKE or equality. Exact-match lookup uses HMAC hash columns:

1. Store a deterministic HMAC-SHA256 hash in `jmbgHash` / `oibHash` column (separate `FIELD_HMAC_KEY`)
2. Search is performed on the hash column
3. Full plaintext is decrypted only when displaying to an authorized user

---

6. Password Hashing

Parameter	Value
Algorithm	bcrypt
Cost factor	12 (adaptable upward as hardware improves)
Salt	Automatically generated by bcrypt library (16 bytes)
Minimum password entropy	8 chars, 1 uppercase, 1 number, 1 special character
Breach check	HaveIBeenPwned API (k-anonymity — only first 5 chars of SHA1 hash sent)

Never: Store plaintext passwords, use MD5 or SHA1 for passwords, use bcrypt cost factor below 10.

---

7. JWT Signing

Parameter	Value
Algorithm	RS256 (RSA + SHA-256) — asymmetric
Private key	2048-bit RSA, stored in `JWT_PRIVATE_KEY` Railway secret
Public key	Stored in `JWT_PUBLIC_KEY` Railway secret (for verification)
Access token lifetime	15 minutes
Refresh token lifetime	7 days
Key rotation	Annually or on compromise

Why RS256 over HS256: RS256 allows future token verification by external services without sharing the signing secret.

---

8. Monetary Data Integrity

Financial amounts must never be stored as floating-point types due to rounding errors in financial calculations.

Parameter	Standard
Database type	`NUMERIC(19,4)` — PostgreSQL exact decimal
Application type	`Decimal.js` library — not JavaScript `number`
Rounding	Banker's rounding (round half to even)
Currency storage	ISO 4217 code (RSD, BAM, EUR) in separate column

---

9. Key Management Summary

Keys are managed per the Key Management Policy (key-management-policy.md). Encryption keys must never be:

- Committed to source code repositories
- Logged in application logs
- Sent over unencrypted channels
- Stored outside Railway environment variables or Vaultwarden

---

10. Prohibited Algorithms

Algorithm	Status	Reason
MD5	PROHIBITED	Cryptographically broken
SHA-1	PROHIBITED for signing/hashing sensitive data	Collision attacks demonstrated
DES / 3DES	PROHIBITED	Insufficient key length
RC4	PROHIBITED	Multiple vulnerabilities
RSA with key < 2048 bits	PROHIBITED	Insufficient for current threat model
AES-128	PERMITTED (not preferred)	AES-256 required for L4 Restricted data
AES-256-CBC	PERMITTED with caution	GCM preferred (provides authentication)
AES-256-GCM	**REQUIRED** for L4 fields	Authenticated encryption

Approval

Role	Name	Signature	Date
Author	CTO		2026-02-23
Reviewer (DPO)			
Reviewer (Engineering Lead)			
Approver	CEO		

Revision #4
Created 2026-02-24 22:50:53 UTC by John
Updated 2026-05-31 20:03:57 UTC by John