healthOS

SECURITY POLICY

Encryption and Key Management

The algorithms in use, encryption in transit and at rest, and how keys are generated, stored, rotated, revoked and recovered.

Edition November 2026 · Public edition · Security: security@healthos.net.au

Contents

Approved algorithms 2

Encryption in transit (requirement 3) 3

Encryption at rest (requirement 4) 3

Key management (requirement 5) 4

End users' PRODA credentials 5

This is the policy. The procedure for rotating each individual key is key-rotation.md, and the design of the signing secrets is jwt-secrets.md; this document does not repeat them.

Approved algorithms

Only algorithms the ISM approves are used. The NDIA names three categories, and this is what the system uses in each.

CATEGORY	USED FOR	ALGORITHM	WHERE
Symmetric	Stored credentials and secrets (field encryption)	AES-256-GCM	`infrastructure/src/crypto/field-encryption.ts`, `api-gateway/src/utils/secrets-key.ts`
Symmetric	The licensed grouper bundle	AES-256-GCM	`reference_data.ardrg_bundles`
Symmetric	Database, backups, Blob storage, Key Vault contents	AES-256, by the Azure platform	Azure storage service encryption
Asymmetric	The SMART on FHIR OIDC `id_token`	RS256 (RSA with SHA-256)	`routes/smart/oidc-keys.ts`
Asymmetric	PRODA device keys, NASH and HI Service certificates	As issued by Services Australia and the Agency	Stored encrypted in the database

CATEGORY	USED FOR	ALGORITHM	WHERE
Asymmetric	TLS server certificates	As issued by the certificate authority	Bound to Azure Container Apps ingress. Checked 22 November 2026: `healthos.net.au` and `app.healthos.net.au` carry DigiCert (GeoTrust TLS RSA CA G1) certificates; the `*.azurecontainerapps.io` hostnames carry Microsoft (Microsoft TLS G2 RSA CA) certificates. All are RSA 2048 signed with SHA-256
Hashing	Passwords	bcrypt, cost 13	`shared/src/auth/password-hash.ts`
Hashing	Token signing, HMAC derivation, integrity	SHA-256 family (HMAC-SHA-256)	`shared/src/auth/service-jwt.ts`

Not used, and not to be introduced: MD5 or SHA-1 for any security purpose, DES or 3DES, RC4, RSA below 2048 bits, ECB mode, or any home-made scheme.

Encryption in transit (requirement 3)

- All traffic between a user and HealthOS is HTTPS. The public entry point is Azure Container Apps ingress, which terminates TLS, and nginx sends HSTS so a browser will not fall back to HTTP.
- SSL in every version, TLS 1.0 and TLS 1.1 are not accepted. The minimum is TLS 1.2, set explicitly where the platform exposes the setting (`minimumTlsVersion` on storage and Redis in `infra/modules/`).
- **TLS 1.3** is negotiated at the public entry point. A handshake against the primary stack's staff hostname on 22 October 2026 used TLS 1.3 with `TLS_AES_256_GCM_SHA384` (`test-results/b2g-conformance/runs/20261024T223611/foundational-evidence/tc-0058-tls.txt`). The test was repeated on 22 November 2026 for each public hostname: `healthos.net.au`, the SaaS stack's `app.healthos.net.au` and its `healthos-staff.calmwater-722d966b` ingress host, and the clinical demo stack's `healthos-staff.purpledesert-2ec61873` host. Each negotiated TLS 1.3 with `TLS_AES_256_GCM_SHA384`, accepted TLS 1.2, and refused TLS 1.0 and 1.1 even with the client's own security level lowered, so the refusal is the server's (`test-results/tls-evidence/20261122/`, one file per host).
- Connections to the database and to Redis are TLS. Calls out to government services (PRODA, Services Australia, the My Health Record gateway, and the NDIA APIs) are HTTPS to an allowlisted set of hostnames (`SA_ALLOWED_HOSTS`, `B2G_ALLOWED_HOSTS`), and no redirect is followed.

NDIA evidence wanted: the software stack used for TLS, the certificates, HTTPS shown enforced, a call to the API, and the handshake. The handshake test above produces most of it.

Encryption at rest (requirement 4)

Every repository that holds participant-related information is encrypted at rest.

STORE	ENCRYPTION	EVIDENCE TO CAPTURE
Azure Database for PostgreSQL (all application data)	AES-256 by the platform, on data and backups	Portal screenshot of the server's data encryption blade
Azure Blob Storage (documents, harness files, logical backups)	AES-256 by the platform	Portal screenshot of the account's encryption blade
Azure Key Vault (secrets)	Encrypted by the platform	Portal screenshot of the vault
Log Analytics (application logs)	Encrypted by the platform	Microsoft's service documentation
Stored credentials inside the database	AES-256-GCM by the application, on top of the platform's encryption	`field-key-sweep.ts` `TARGETS` lists every column

The keys for platform encryption are Microsoft-managed. Moving to customer-managed keys is not planned (decision, 21 Nov 2026): they add operational risk and cost with no current requirement, and are revisited if a customer contract requires them.

Key management (requirement 5)

The NDIA lists the lifecycle stages a key management policy must cover. Each is answered here.

Generation. Keys and secrets are generated with a cryptographically secure random source (`crypto.randomBytes` or `openssl rand`), never chosen by a person and never derived from a word. `deploy-infra.sh init` generates a stack's secrets when the stack is created.

Length and complexity.

KEY	MINIMUM
Field and credential encryption keys	256 bits (32 random bytes)
JWT signing secrets	48 random bytes; the services refuse to start in production on a missing, short or reused one
Database and Redis passwords	32 random bytes
RSA keys	2048 bits

Distribution. A secret moves from where it is generated into Azure Key Vault by the infrastructure deployment, and from Key Vault into a container as a Container Apps secret reference. It is never sent by email or chat, never committed, and never written to a log: Pino redacts tokens, keys and passwords at the logger (`utils/logging/redaction.ts`).

Storage. Deployment secrets are in the stack's Key Vault. Secrets the application is handed at runtime are in a separate API vault, so the application cannot overwrite the secrets the platform runs on. Each stack has its own vaults; no key is shared between stacks. The `.env.infra.<id>` files that feed a deployment are gitignored. They are held only on the owner's workstation, which uses full-disk encryption, with a copy in the password manager vault that the sealed emergency access pack opens (risk R-014).

Access. The application reads its secrets through its managed identity, with the Key Vault Secrets User role on the deployment vault. People reach a vault only through a named Azure account with a role granted under backup-restoration-and-privileged-access.md. Each use of a government credential by the application is written to `credential_access_log`.

Rotation and renewal. Every key is rotated at least annually; the field encryption key every 90 days; any key at once when someone with access leaves or the key may have been exposed. Certificates and PRODA device keys are renewed before they expire, and the application shows their expiry. The steps for each are in key-rotation.md, and each rotation is recorded with its date, the key, who did it and the stack.

Revocation. A key is revoked by replacing it and disabling the old version in Key Vault. A credential issued by a third party (a PRODA device, a NASH certificate, an Entra or Stripe secret) is revoked with its issuer first, so the old one stops working everywhere.

Archiving. During a rotation the previous encryption key is kept, as `FIELD_ENCRYPTION_KEY_PREVIOUS`, only until the re-encryption sweep reports nothing left under it; it is then removed. Old versions in Key Vault are disabled, not left active. No encryption key is archived outside Key Vault.

Destruction of compromised keys. A key that may have been exposed is rotated immediately and its old versions disabled once the re-encryption sweep is clean. Purge protection is on, so Key Vault removes a deleted version itself after the 90-day retention; it cannot be purged sooner. Exposure of a key is exposure of what it protected: it is handled as a security incident and assessed as a data breach under Security Incident and Data Breach Response, whatever the rotation achieves.

Recovery. Every vault has soft delete (90 days) and purge protection on (`infra/modules/keyvault*.bicep`), so a deleted secret can be recovered and cannot be purged early. If the field encryption key is lost with no copy, the credentials it protects cannot be recovered and must be re-entered; the database itself is unaffected, because it is encrypted by the platform. The recovery copy of each stack's secrets is the password manager vault, whose emergency kit is held sealed under risk R-014 (see the continuity policy, section 7).

End users' PRODA credentials

The NDIA API Terms (cl. 2.5(c)) require that a provider's PRODA credentials are entered in a way that does not let HealthOS or its staff access them.

Owner's decision (21 Nov 2026): holding the encrypted device key on the provider's behalf is accepted as meeting this clause. No confirmation is sought from the Digital Partnership Office.

healthOS

Questions about security

Email **security@healthos.net.au**. To report a vulnerability, see https://healthos.net.au/trust/#disclosure.

© 2026 HealthOS Pty Ltd · ABN 81 697 460 889. This is the public edition. The web edition is at https://healthos.net.au/trust/policy-encryption-and-keys.html.