SecureVault Encryption Overview

Document ID: PRD-SV-005 **Last Updated:** 2024-02-10 **Owner:** SecureVault Engineering **Classification:** Public

Introduction

SecureVault uses industry-standard encryption to protect your secrets at rest and in transit. This document explains our encryption architecture, key management, and security practices.

Encryption Architecture

Defense in Depth

```
           Transport Layer
         TLS 1.3 / mTLS

          Application Layer
    Client-side encryption (optional)

           Storage Layer
         AES-256-GCM

       Key Encryption Layer
    Master Key + Key Hierarchy
```

Encryption at Rest

Secret Encryption

All secrets are encrypted using **AES-256-GCM** (Galois/Counter Mode).

Properties: - 256-bit key strength - Authenticated encryption (prevents tampering) - Unique IV per encryption operation - Associated data for context binding

Process: 1. Generate unique Data Encryption Key (DEK) per secret 2. Encrypt secret with DEK using AES-256-GCM 3. Encrypt DEK with Key Encryption Key (KEK) 4. Store encrypted secret + encrypted DEK

Key Hierarchy

```
                Master Key



        KEK-1     KEK-2     KEK-3


    DEK DEK DEK DEK DEK DEK DEK DEK DEK
```

Key Types: | Key | Purpose | Rotation | |—–|———–|———–-| | Master Key | Protects KEKs | Yearly | | Key Encryption Key (KEK) | Protects DEKs | Quarterly | | Data Encryption Key (DEK) | Encrypts secrets | Per-secret |

Master Key Protection

Standard (Cloud-hosted): - Master key split using Shamir's Secret Sharing - Requires 3 of 5 key shares to reconstruct - Shares distributed to executive custodians - Hardware Security Module (HSM) storage

Enterprise (BYOK - Bring Your Own Key): - Customer provides master key material - Supported HSMs: AWS CloudHSM, Azure Dedicated HSM, Google Cloud HSM - Key never leaves customer HSM

Encryption in Transit

TLS Configuration

All API communications use **TLS 1.3**.

Supported Cipher Suites: - TLS_AES_256_GCM_SHA384 - TLS_CHACHA20_POLY1305_SHA256 - TLS_AES_128_GCM_SHA256

Disabled: - TLS 1.0, 1.1, 1.2 (deprecated) - All CBC mode ciphers - All export ciphers

Certificate Management

- Certificates from trusted CA (DigiCert)
- Certificate pinning available for mobile/desktop
- Automatic certificate rotation
- HSTS enabled (max-age: 1 year)

Service-to-Service (mTLS)

Internal services use mutual TLS: - Client certificates required - Certificate rotation every 24 hours - Automatic certificate provisioning via internal CA

Client-Side Encryption

For maximum security, encrypt secrets before sending to SecureVault.

How It Works

```
Client                          SecureVault

      1. Generate local key
      2. Encrypt secret locally
      3. Send encrypted secret   >
                                      4. Encrypt again (envelope)
                                      5. Store doubly-encrypted

      6. Request secret         <
                                      7. Return envelope-encrypted
      8. Decrypt locally
```

Benefits

- SecureVault never sees plaintext
- Protection against insider threats
- Meets strict compliance requirements

SDK Support

```
from securevault import Client, LocalEncryption
```

```python
# Initialize with local encryption
vault = Client(
    token="sv_xxx",
    local_encryption=LocalEncryption(
        key_path="/path/to/local.key"
    )
)

# Secrets are encrypted locally before upload
vault.secrets.create("api-key", "super-secret-value")
```

Key Rotation

Automatic Rotation

DEK Rotation: - New DEK generated on every secret update - Old versions retained (configurable) - No service interruption

KEK Rotation: - Quarterly automatic rotation - Re-encrypts all DEKs - Rolling update, no downtime

Manual Rotation

Force immediate key rotation:

```bash
securevault rotate-keys --scope project --project-id proj_123
```

Rotation Audit

All key rotations are logged: - Rotation timestamp - Previous key identifier - New key identifier - Initiator (system/user)

Cryptographic Standards

Algorithms Used

Purpose	Algorithm	Standard
Symmetric encryption	AES-256-GCM	NIST SP 800-38D
Key derivation	HKDF-SHA256	RFC 5869
Password hashing	Argon2id	RFC 9106
Digital signatures	Ed25519	RFC 8032
Key agreement	X25519	RFC 7748
Random generation	CSPRNG	NIST SP 800-90A

Compliance

SecureVault encryption meets: - NIST Cryptographic Standards - FIPS 140-2 Level 3 (HSM modules) - SOC 2 Type II - ISO 27001 - GDPR encryption requirements - HIPAA (with BAA) - PCI-DSS

Security Practices

Secret Lifecycle

1. **Creation:** Encrypted immediately, plaintext never logged
2. **Storage:** Encrypted at rest, encrypted backup
3. **Access:** Decrypted in memory only, short-lived
4. **Rotation:** Seamless re-encryption
5. **Deletion:** Secure wipe, key destruction

Memory Protection

- Secrets held in memory only during active use
- Memory scrubbed after use
- No swap file exposure
- Protected memory regions where available

Audit Logging

All cryptographic operations logged: - Secret access (read/write) - Key operations - Administrative changes - Authentication events

Logs are: - Encrypted - Tamper-evident - Retained per policy (default 2 years)

Disaster Recovery

Backup Encryption

- Backups encrypted with separate backup key
- Backup keys escrowed securely
- Geographic distribution of key shares
- Regular recovery testing

Key Recovery

In case of key loss: 1. Assemble key custodians (3 of 5) 2. Reconstruct master key in secure environment 3. Re-initialize key hierarchy 4. Validate secret accessibility

---

Technical Specifications

Performance

Operation	Latency (p99)
Secret read	< 50ms
Secret write	< 100ms
Key rotation	< 5s per 1000 secrets
Bulk encryption	10,000+ ops/sec

Limits

Limit	Value
Max secret size	64 KB
Max key size	4 KB
Versions per secret	100 (configurable)
Concurrent operations	10,000+

FAQ

Is my data encrypted at rest?

Yes, all data is encrypted using AES-256-GCM with unique keys per secret.

Can SecureVault employees access my secrets?

No. Secrets are encrypted with keys we don't have access to. With client-side encryption, we never see plaintext.

What happens if encryption keys are compromised?

We maintain defense in depth. Key hierarchy limits exposure. Compromise of one key doesn't expose all secrets.

How do I verify encryption is working?

Use our verification API or audit logs to confirm encryption status.

Related Documents: Setup & Installation (PRD-SV-001), Security Whitepaper (PRD-SV-020), Compliance Guide (PRD-SV-025)