CRYPTOGRAPHY POLICY

Information Security Program

Version 1.0

2026-05-22

Prepared for Walker-Miller Energy Services

Project Baseline, Inc. | Colorado | EIN 27-0639457 | todd@project-baseline.com | project-baseline.com

Entity: Project Baseline, Inc.

Effective date: 2026-05-22

Version: 1.0

Owner: Todd Walton, Information Security Officer

Review cadence: Annual; off-cycle on cryptographic standard changes

1. PURPOSE

This Cryptography Policy defines the cryptographic standards Project Baseline, Inc. (PB) uses to protect data in transit, at rest, and in use across PB systems. It exists to ensure consistent application of modern cryptography, to provide a documented standard for client and auditor review, and to give Todd a clear operational baseline.

2. SCOPE

All PB systems handling data including:

- Public-facing web properties (TLS configuration)
- VPS-hosted databases and files (encryption at rest where applicable)
- Endpoint devices (full-disk encryption)
- Authentication (password hashing)
- API authentication (token signing, secret management)
- Backup data (backup encryption)

3. CRYPTOGRAPHIC STANDARDS

3.1 DATA IN TRANSIT (TLS)

PROPERTY	STANDARD
Minimum TLS version	TLS 1.2
Preferred TLS version	TLS 1.3
Prohibited	SSL 2.0, SSL 3.0, TLS 1.0, TLS 1.1
Cipher suites	Forward-secret AEAD ciphers only (ECDHE + AES-GCM or CHACHA20-POLY1305)
Certificate authority	Let's Encrypt (free, automated via Certbot)
Certificate algorithm	ECDSA preferred (P-256), RSA 2048+ acceptable
HSTS	Enabled on all public domains via Cloudflare or nginx
Certificate renewal	Automated via Certbot; manual verification quarterly

Implementation:

- Cloudflare-fronted domains: TLS configured at Cloudflare (Full Strict mode)
- nginx-direct domains (currently only walkermillertad.project-baseline.com; remediation in Section F adds Cloudflare): TLS 1.2 minimum enforced in nginx config
- Internal service-to-service on VPS (gunicorn behind nginx on 127.0.0.1): plaintext acceptable because traffic does not leave the localhost interface

3.2 DATA AT REST

STORAGE	ENCRYPTION	NOTES
Google Workspace (Gmail, Drive, Docs)	AES-256 (Google-managed)	Provider-managed encryption
DigitalOcean VPS filesystem	Provider-managed encryption at the droplet boundary	DigitalOcean encrypts droplet storage; documented in DigitalOcean security overview
SQLite databases (`_tad_abuse_guards.db`, `nonprofit-portal/db/database.sqlite`)	Filesystem-level only (relies on VPS encryption)	**Gap acknowledged:** No application-layer encryption. Acceptable risk given (a) the data is low-sensitivity (abuse-control metadata + nonprofit profile data), (b) VPS is single-tenant under PB control, (c) backup encryption is enforced. Re-evaluate if data sensitivity changes.
Uploaded files (`nonprofit-portal/uploads/`)	Filesystem-level only	Same rationale as SQLite. MIME validation and access controls in application layer.
Todd's Windows laptop	BitLocker (AES-128 or AES-256 depending on platform default; AES-256 preferred)	To be verified in Section F. Mandatory.
Off-VPS backups	AES-256 with PB-controlled key	Per Backup Policy. Encryption key stored in PB password manager, separate from the backup storage location.
Stripe payment data	Provider-managed (PCI-DSS Level 1)	PB never stores card data

3.3 PASSWORD HASHING

USE CASE	ALGORITHM	CONFIGURATION
Nonprofit-portal user passwords	bcrypt	Cost factor 12 (current implementation)
Any future application requiring password storage	bcrypt (cost 12+), argon2id (preferred for new), or scrypt	Pick one per app; document choice

Prohibited: MD5, SHA-1, SHA-256 single-pass, plain text. No PB system stores passwords in any prohibited form.

3.4 API TOKEN AND SECRET SIGNING

USE CASE	ALGORITHM
cos-webhook authentication (`x-cos-secret` header)	Shared secret comparison via constant-time check
Stripe webhook signature verification	HMAC-SHA256 (Stripe-provided signing secret)
Cookie session tokens (if used)	HMAC-SHA256 with PB-controlled signing key
JWT (if any current or future use)	RS256 or ES256 (asymmetric); HS256 acceptable for internal-only with strong shared secret

Prohibited: Unsigned tokens. Tokens with secrets stored in source control.

3.5 RANDOM NUMBER GENERATION

All cryptographic randomness uses platform-native CSPRNG:

- Python: `secrets` module
- Node.js: `crypto.randomBytes()`
- Shell: `/dev/urandom`

Prohibited: `Math.random()`, `random.random()`, or any non-cryptographic RNG for security-relevant values (tokens, salts, nonces, session IDs).

4. KEY MANAGEMENT

4.1 TLS PRIVATE KEYS

- Generated by Let's Encrypt automation
- Stored on VPS at `/etc/letsencrypt/live/[domain]/privkey.pem`
- File permissions: 0600, root-owned
- Rotated automatically every 90 days by Certbot

4.2 APPLICATION SIGNING KEYS (HMAC SECRETS, COOKIE SECRETS, WEBHOOK SECRETS)

- Stored in `.env` files on VPS and Todd's laptop ONLY

- Never committed to GitHub
- Rotated at minimum annually OR on any suspected compromise
- Inventory maintained per Access Control Policy Section 10

4.3 BACKUP ENCRYPTION KEYS

- Generated using platform-native CSPRNG, 256-bit
- Stored in PB password manager (separate from backup storage location)
- Documented recovery procedure ensures Todd can decrypt backups even if primary password manager is unavailable (secondary copy in sealed envelope or secondary password manager)
- Rotated annually

4.4 API KEYS (ANTHROPIC, OPENAI, STRIPE, CLOUDFLARE, ETC.)

- Stored in `.env` files only
- Rotated annually OR on any suspected compromise OR on contractor offboarding (if any contractor had access)
- Per Access Control Policy Section 10, inventoried with provider, purpose, environment, creation date, last rotation, next scheduled rotation

5. CRYPTOGRAPHIC FAILURE RESPONSE

If a cryptographic primitive PB uses is deprecated or broken (for example, bcrypt cost 12 becomes insufficient, or a TLS vulnerability is disclosed):

1. Detect via vendor notifications, security advisories, or monitoring
2. Assess scope (which systems are affected, what data is exposed)
3. Implement mitigation per Incident Response Plan
4. Update this policy with the new standard
5. Migrate all affected systems to the new standard within 90 days
6. Document migration in Risk Register

6. SPECIFIC IMPLEMENTATION AUDITS

The following items will be verified or implemented in Section F:

1. nginx TLS configuration on walkermillertad.project-baseline.com confirms TLS 1.2 minimum (and remediation moves it behind Cloudflare for additional protection)
2. BitLocker is enabled on Todd's Windows laptop (`manage-bde -status C:` returns Protection On)
3. SSH key permissions on `~/.ssh/vps_key` are 0600
4. nonprofit-portal bcrypt cost factor confirmed at 12 (verify in `routes/auth.js`)
5. Stripe webhook signature verification implemented correctly in `routes/webhooks.js`
6. Backup encryption is in place (encrypted at rest at the backup destination)
7. No API keys committed to GitHub history (gitleaks scan)

7. ACKNOWLEDGMENT

Todd Walton acknowledges he has read and agrees to operate in accordance with this Cryptography Policy.

Signature: Todd Walton, Principal, Project Baseline, Inc.

Date: 2026-05-22

END OF DOCUMENT