BONAFIDE™ SPECIFICATION V1.0 — PART 2

Cryptographic Foundation

Composable root derivation, root secret gesture system, biometric processing, key hierarchy, data encryption, Merkle integrity, and algorithm selection

1. Purpose

This part defines the cryptographic foundation of the Bonafide specification: how the Bio Root is derived, how keys are generated through the security level hierarchy, how data is encrypted, and why the system's stateless properties provide security guarantees that no password-based system can match.

2. Composable Root Derivation

2.1 The Bio Root

The Bio Root is the cryptographic root of the user's vault. Every key in the vault hierarchy is ultimately derived from the Bio Root. The Bio Root is computed—never stored—during each authentication event and destroyed immediately after key derivation completes.

The Bio Root is derived from three composable inputs:

Bio Root = **Hash(biometric || root_secret_hash || manual_passphrase)**

- **biometric:** The user's multi-modal biometric input, processed entirely within the device's secure element. Never transmitted, never stored. Required for every authentication.
- **root_secret_hash:** A hash derived from one or more user-chosen authentication gestures, set once during initial vault creation and stored in the device's hardware enclave. Automatically appended to every authentication—the user does not re-enter it daily. Required for new device enrollment.
- **manual_passphrase:** An optional input typed or gestured at authentication time. When present, it selects an alternate persona (Part 7). When absent (empty string), it selects the default persona.

2.2 Stateless Derivation

The derivation is stateless: there is no stored "correct answer" to compare against. The Hash function always produces a 256-bit key regardless of inputs. The resulting key either decrypts vault data or it does not. The system never indicates whether the derivation was "correct"—there is no error message, no timing difference, no observable signal.

An attacker who attempts authentication with incorrect inputs receives a valid-looking 256-bit key that decrypts nothing. They cannot distinguish "wrong biometric" from "wrong root secret" from "wrong passphrase" from "this person has no vault." Every combination produces a plausible key. The attacker is searching in the dark with no oracle.

This property is unique to Bonafide. Every other authentication system—passwords, TOTP, FIDO2, even most biometric systems—provides an oracle (login succeeds or fails). Bonafide provides silence.

3. Root Secret Gesture System

3.1 What the Root Secret Is

The root secret is a user-chosen authentication input that strengthens the Bio Root beyond biometric alone. It is set once during initial vault creation, hashed, and the hash is stored in the device's hardware enclave. On subsequent authentications, the enclave automatically appends the stored hash—the user never re-enters it. The root secret makes biometric spoofing on a foreign device useless: even a perfect biometric clone produces the wrong Bio Root without the root secret.

3.2 Gesture Categories

The root secret is not limited to a typed passphrase. It is derived from one or more authentication gestures chosen by the user:

Physiological Biometric Gestures

- A specific finger not used for daily unlock (ring finger, pinky, thumb on opposite hand)
- A specific facial expression (tongue out, wink, puffed cheeks, raised eyebrow)
- A specific spoken word or phrase (the vocal signature on a specific utterance, not speech-to-text)
- A specific eye movement pattern (look left, right, up, down in a chosen sequence)
- A specific hand gesture to camera (peace sign, three fingers, fist, open palm)

Behavioral Biometric Gestures

- Keystroke dynamics on a specific sequence (typing "aaa" but the inter-key timing is the unique signature)
- A specific drawing or signature on touchscreen (pressure, speed, and acceleration, not the shape)
- A specific tap pattern on the device (rhythm, pressure, location)
- A specific device movement (shake pattern, tilt sequence, rotation gesture)

Knowledge Gestures

- Traditional text passphrase
- PIN or pattern
- Selection from a visual grid (pick your three images from a set)

3.3 Composability

The user can choose one gesture or combine several. Each gesture is processed into a stable hash by the enclave. The hashes are concatenated and hashed together to produce root_secret_hash:

root_secret_hash = Hash(gesture_1_hash || gesture_2_hash || ... || gesture_n_hash)

The number and type of gestures are the user's choice. The system does not reveal how many gestures were combined or what types were used. An attacker faces a combinatorial explosion before they can even begin: which gesture types? How many? In what order? What specific instance of each?

3.4 Stability and Fuzzy Extraction

Gestures that involve biometric input (facial expression, voice, keystroke timing) are inherently noisy—the user will not reproduce them identically each time. The enclave uses fuzzy extraction (the same technique used for primary biometric processing) to derive a stable hash from noisy input. The fuzzy extractor stores a public helper value that enables recovery of the same hash from approximately correct input without revealing the input itself.

Gesture-specific tolerance parameters are defined per gesture type. Keystroke timing tolerates ±50ms per keystroke. Facial expression recognition tolerates natural variation in muscle movement. Voice print extraction uses established speaker verification algorithms with configurable confidence thresholds.

3.5 New Device Enrollment

When a user enrolls a new device, they must provide the root secret gestures manually—the new device does not have root_secret_hash in its enclave. The user authenticates biometrically and performs the root secret gestures. The new device derives root_secret_hash, computes the Bio Root, and verifies it against the vault. Upon success, root_secret_hash is stored in the new device's enclave for future automatic use.

4. Biometric Processing

4.1 Multi-Modal Fusion

The Bonafide specification requires a minimum of two distinct biometric modalities for Bio Root computation on vault holder devices. Modalities are classified by strength:

- **Strong:** Ultrasonic fingerprint, 3D face (structured light/ToF), iris scan. High distinctiveness, strong liveness detection.
- **Medium:** Optical fingerprint, 2D face with IR liveness, voice with anti-replay.
- **Weak:** 2D camera face (no IR), basic voice, behavioral biometrics alone.

A vault holder must present at least one strong or two medium modalities. Devices with only weak modalities cannot serve as vault holders (Part 10).

4.2 Biometric Template Processing

Biometric data is processed entirely within the device's secure element:

- Raw biometric data (fingerprint image, face scan, iris pattern, voice sample) is captured by the device sensor.
- The secure element extracts a template (compact mathematical representation) from the raw data.
- The raw data is destroyed immediately after template extraction. It is never stored.
- The template is processed through a fuzzy extractor to produce a stable biometric hash.
- The biometric hash is combined with root_secret_hash and manual_passphrase to derive the Bio Root.
- The template is destroyed after the Bio Root is derived. No biometric data persists after authentication.

4.3 Liveness Detection

Every biometric modality must include liveness detection to prevent spoofing:

- Fingerprint: ultrasonic sensors detect sub-surface features (blood flow, tissue layers) that printed or silicone replicas cannot reproduce.
- Face: structured light or time-of-flight depth mapping detects 3D geometry that photos and screens cannot reproduce. IR liveness detects thermal signatures.
- Iris: near-infrared illumination detects pupil response and tissue characteristics.
- Voice: anti-replay analysis detects recordings. Challenge-response requires speaking a random phrase.

5. Key Hierarchy

5.1 Derivation Chain

The Bio Root is the top of the key hierarchy. All lower keys are derived from it using HKDF (HMAC-based Key Derivation Function) with context-specific parameters at each level:

Level 0: Bio Root = Hash(biometric || root_secret_hash || manual_passphrase)

Level 1: Branch Key = HKDF(Bio Root, institution_id || branch_id)

Level 2: Channel Key = HKDF(Branch Key, channel_id)

Level N: DEK = HKDF(Parent Key, quantum_id || security_level)

Each level adds context-specific entropy. A branch key derived for Chase Bank cannot access the branch at Mount Sinai Hospital because the institution_id differs. A channel key derived for transactions cannot access the statements channel. A DEK derived at Level 5 cannot decrypt a quantum at Level 7 because the security_level parameter differs.

5.2 Security Level Key Depth

Higher security levels require deeper key derivation chains. A Level 3 quantum’s DEK is derived through 3 HKDF rounds. A Level 7 quantum’s DEK is derived through 7 rounds. Each round adds computational cost and each round requires the enclave to hold intermediate state—which is why higher security levels require higher enclave tiers (Part 10). A Tier 3 (TPM) enclave cannot efficiently perform 7 rounds of HKDF with secure intermediate storage; a Tier 1 (dedicated SE) enclave can.

5.3 Key Ephemerality

No key in the hierarchy is permanently stored except root_secret_hash (in the device enclave). The Bio Root is computed during authentication and destroyed after key derivation. Branch keys, channel keys, and DEKs are derived on demand, used for the operation, and destroyed. Session keys are time-bounded. The only persistent cryptographic material in the entire system is root_secret_hash in the enclave and the encrypted DEK wrappings on disk.

6. Data Encryption

6.1 Quantum Encryption

Every quantum payload is encrypted with AES-256-GCM using a quantum-specific DEK. The DEK is never reused across quanta. AES-256-GCM provides both confidentiality (encryption) and integrity (authentication tag). Any modification to the ciphertext is detectable.

6.2 DEK Wrapping

The DEK is wrapped (encrypted) independently for each authorized key path:

- User wrapping: DEK encrypted with the user’s derived key at the appropriate level. Always present.
- Institutional wrapping: DEK encrypted with the institution’s branch-derived key. Present during active peering.
- Third-party wrapping(s): DEK encrypted with purpose-specific third-party keys. Present only for active authorizations.

Each wrapping is a separate ciphertext stored alongside the encrypted quantum. Revoking an institutional wrapping means deleting one ciphertext and re-encrypting the DEK without it. The quantum payload is not re-encrypted—only the DEK wrapping changes. This makes revocation fast (small ciphertext operation) regardless of quantum payload size.

6.3 Session Keys

During an active session, the runtime derives session keys from the appropriate key path (user, institutional, or third-party) combined with session-specific parameters (session nonce, timestamp, device attestation). Session keys are used for the transport layer—encrypting API requests and responses between the client and the runtime. Session keys expire with the session and are not reusable.

6.4 Dual-Derived Session Tokens

For institutional sessions, the session token is dual-derived: the user’s half is generated during biometric authentication, the institution’s half is generated by the runtime. Both halves must be present to form a valid session token. This prevents either party from unilaterally creating sessions and ensures that every session is a mutual agreement between the user and the institution.

7. Merkle Integrity

7.1 Quantum Integrity

Each quantum’s encrypted payload is hashed to produce a Merkle leaf. The leaves are organized into a Merkle tree per branch. The branch’s Merkle root is a single hash that verifies the integrity of every quantum in the branch. Modifying, inserting, or deleting any quantum changes the root.

7.2 Ledger Integrity

The immutable ledger (Part 1, Section 8) is a separate Merkle tree. Each ledger entry is hashed and chained. The ledger root verifies the complete operation history. Ledger roots are periodically submitted to the blind validation network (Part 5) for independent verification.

7.3 Tombstoning for Erasure

When a quantum is deleted (e.g., GDPR right-to-erasure), the DEK is destroyed and the quantum’s Merkle leaf is replaced with a tombstone marker. The tombstone preserves the tree’s structure and the Merkle root’s verifiability. The data is cryptographically destroyed (the DEK no longer exists), but proof of former existence remains. This satisfies GDPR’s actual requirement: the data is gone, the deletion is provable.

8. Cryptographic Algorithms

8.1 Algorithm Selection

The specification mandates the following algorithms for the reference implementation. Deployments may substitute equivalent algorithms of equal or greater strength if required by jurisdictional regulations (e.g., GOST algorithms for Russian institutional deployments under the Regulated Profile).

Function	Algorithm	Key Size	Notes
Bio Root derivation	SHA-512 with HKDF	256-bit output	Truncated from 512-bit hash
Key derivation	HKDF-SHA-256 (RFC 5869)	256-bit	Context parameters prevent cross-domain key reuse
Symmetric encryption	AES-256-GCM	256-bit key, 96-bit nonce	Authenticated encryption with associated data
Merkle tree hashing	SHA-256	256-bit	Collision resistance sufficient for integrity verification
Fuzzy extraction	Secure sketch + strong extractor	Configurable	Per-modality parameters defined in implementation guide
Key wrapping	AES-256-KW (RFC 3394)	256-bit	Wrapping DEKs for each key path
Digital signatures	Ed25519 or ECDSA P-384	256 or 384 bit	Device attestation, ledger entries, validator consensus
TLS	TLS 1.3 with AEAD	256-bit	CHACHA20-POLY1305 or AES-256-GCM

8.2 Algorithm Agility

The specification supports algorithm agility. Each quantum's metadata includes the algorithm suite used for encryption. This allows the ecosystem to migrate to post-quantum algorithms when standardized without requiring a flag-day transition. Existing quanta remain readable with the original algorithm suite while new quanta use the updated suite. Key rotation migrates individual quanta to the new suite over time.

8.3 Post-Quantum Readiness

The current algorithm suite is not post-quantum resistant. When NIST's post-quantum cryptographic standards reach deployment maturity (ML-KEM, ML-DSA, SLH-DSA), the specification will publish an updated algorithm suite. The key hierarchy and derivation model are algorithm-agnostic—only the underlying primitives change. The vault structure, quantum model, and access control architecture are unaffected by algorithm migration.

Bonafide™ — Privacy by architecture, not by promise.

An open specification by Sly Technologies Inc. | bonafide.id | bonafideid.org
V1.0 — February 2026