What is claimed is:

1. A computer-implemented method for memory-native two-stage authentication, comprising:
   generating, by a sender agent, a dynamic agent hash (DAH_t) as a successor of a prior trusted dynamic agent hash (DAH_{t−1}) generated by the sender agent, wherein the DAH_t is generated under an update rule that incorporates at least one unpredictability contribution and a volatile salt;
   deriving, by the sender agent, a symmetric encryption key from a current dynamic identity of a recipient selected from a recipient dynamic agent hash (DAH_R) or a recipient dynamic device hash (DDH_R);
   encrypting a payload with the symmetric encryption key and embedding within the encrypted payload an embedded sender dynamic agent hash (DAH_S) computed contemporaneously with the DAH_t;
   constructing, by the sender agent, a message comprising a transport header and the encrypted payload, and placing the DAH_t in the transport header and the DAH_S within the encrypted payload, wherein the message does not include the symmetric encryption key;
   transmitting, by the sender agent, the message to the recipient;
   receiving, by the recipient, the transmitted message and reconstructing, from a locally retained trust-slope state for the sender agent that includes at least the DAH_{t−1} most recently validated and previously accepted by the recipient, an expected successor candidate for time t under the update rule and within a recipient-defined set of policy-bounded continuity parameters;
   validating, by the recipient, the DAH_t against an expected successor candidate;
   deriving, by the recipient, a recipient symmetric encryption key from a corresponding one of DAH_R or DDH_R and decrypting the payload;
   extracting, by the recipient, the DAH_S from the decrypted payload and validating the DAH_S against a reconstructed trust slope for the sender agent obtained by advancing the locally retained trust-slope state under the update rule and within the recipient-defined set of policy-bounded continuity parameters; and
   accepting, by the recipient, the message only upon successful validation of both the DAH_t and the DAH_S.
2. The method of claim 1, wherein the accepting and validating are performed without reliance on persistent private keys or external certificate authorities.

3. The method of claim 1, wherein the unpredictability contribution includes a keyed derivation from a static hardware anchor and a volatile per-epoch salt.

4. The method of claim 1, wherein the unpredictability contribution includes an extractor output over a stability-tuned local state vector, the extractor output being used without exposing a raw local state.

5. The method of claim 1, wherein the update rule includes a hardware-anchor derivation and a local-state extractor output and wherein both the hardware-anchor derivation and the local-state extractor output are concatenated in the update rule.

6. The method of claim 1, further comprising rotating an entropy anchor upon detection of staleness and recording a forward link configured to bind a terminal value of a prior epoch to a new initial identity, and rejecting identifiers from an expired epoch except for bridging proofs that open through the forward link.

7. The method of claim 1, further comprising forecasting a near-term successor identity and an acceptance envelope based on cadence statistics and role-transition models; classifying a presented successor as consistent when the presented successor lies within the acceptance envelope; and degrading trust, requesting supplemental proofs, or quarantining when the presented successor falls outside the acceptance envelope.

8. The method of claim 1, further comprising validating, prior to decryption, header-level continuity of the DAH_t against an expected successor and, after decryption, validating payload-level continuity of the DAH_S against a reconstructed trust slope, and rejecting the message without external registry lookup upon failure of validation of either header-level continuity of the DAH_t or payload-level continuity of the DAH_S.

9. The method of claim 1, wherein the symmetric encryption key is derived via a key derivation function keyed by the DAH_R or DDH_R and a context tag, and wherein no asymmetric key exchange is performed.

10. The method of claim 1, wherein, when the sender agent cannot derive a symmetric key from the DAH_R or DDH_R, deriving, by the sender agent, a provisional key from a last trusted recipient anchor and, upon decryption failure, performing, by the sender agent, a fallback including a checkpoint request that yields a bounded proof window or a short challenge–response rekey handshake.

11. The method of claim 10, further including retrying, by the sender agent, decryption within a policy-bounded attempt window.

12. The method of claim 1, further including separating by domain extractor tokens by a fixed public seed and context tag per deployment, and enforcing by validation an acceptance envelope that rejects off-manifold drift without exposing raw local state vectors.

13. The method of claim 1, further including applying, by the recipient, a two-epoch acceptance window for recipient identity, enforcing per-sender rate limits on failed decryptions, and emitting opaque failure codes to prevent oracle leakage.

14. The method of claim 1, further comprising rotating the DAH_t presented in the header at a policy-defined cadence independent of payload semantics.

15. The method of claim 1, wherein deriving the symmetric key includes performing a key derivation function keyed by the DAH_R or DDH_R and a domain-separated context tag, and wherein the derived key expires with a recipient epoch to prevent cross-epoch decryption.

16. A system for agent mutation entanglement, comprising:

a host device configured to compute a dynamic device hash (DDH_t) as a successor of a prior dynamic device hash (DDH_p) under an update rule that incorporates at least one unpredictability contribution and a volatile salt;

a semantic agent configured to execute on the host device and to compute a successor dynamic agent hash (DAH_s) from a prior dynamic agent hash (DAH_p) and a host mutation token derived from the DDH_t and a mutation class associated with the host device;

an entanglement module configured to emit a signed entanglement trace that records DAH_p, DDH_t, the host mutation token, DAH_p, and mutation metadata; and

a validator configured to accept DAH_s only if the entanglement trace opens to DDH_t under policy and DAH_s is a valid successor of DAH_p.

17. The system of claim 16, wherein the host mutation token comprises a cryptographic hash of DDH_t, mutation class, and epoch information, and the entanglement trace includes a signature of the host device.

18. The system of claim 16, further including a monitoring module configured to detect invalid entanglement, cadence anomaly, neighborhood mismatch of extractor outputs, and stale salt, and to degrade trust-score of the semantic agent or quarantine the semantic agent upon detection of invalid entanglement, cadence anomaly, neighborhood mismatch of extractor outputs, or stale salt.

19. The system of claim 16, wherein the semantic agent includes a policy reference to a policy agent that specifies quorum roles, voting weights, and eligibility for mutation validation, and is

configured to accept entangled mutations only when quorum roles, voting weights, and eligibility for mutation validation are consistent with the policy.

20. The system of claim 16, further including a message authentication code configured to authenticate the entanglement trace by a value derived from DDH_t under a domain-separated key derivation function in lieu of a digital signature, wherein the key is ephemeral and locally scoped to an epoch of the host device.

21. The system of claim 16, wherein the host device is configured to employ an ephemeral signing keypair minted per epoch and destroyed upon rotation, and including a verifier configured to accept the entanglement trace only when an epoch identifier opens to DDH_t under policy.