VERIFICATION INCOMPLETE

Run the Golden Path demo to generate a complete receipt chain with verification.

VERIFICATION CHECKS

Check	Result
Ed25519 Signatures	NOT RUN
Hash Chain Integrity	NOT RUN
Merkle Root (RFC 6962)	NOT RUN
x401 Identity Binding	NOT RUN
Wallet-Signed Anchor	NOT RUN
Overall Verdict	NOT RUN

RECEIPT CHAIN SUMMARY

Field	Value
Total Receipts	0
Payments Approved	0
Payments Denied	0
x401 Identity Bindings	0
Settlement Bindings	0
Signed Artifacts	0
Active Agents	0

AGENT REGISTRY

Each agent has a unique Ed25519 signing key. All artifacts are signed independently - no agent can forge another agent's signature.

Agent	Role	Signing Key (kid)	Artifacts
Coordinator	x402 marketplace discovery + agent routing	coordinator-verigate-8b17c27...	0
Gateway	Deterministic policy eval + signed receipts	gateway-verigate-init...	0
Auditor	EU AI Act / NIST / DORA compliance proof	auditor-verigate-ad0d35ed...	0
Investigator	Forensic evidence + severity classification	investigator-verigate-ef3b2d...	0
Recommender	Circle policy recommendations	recommender-verigate-cddf497...	0
Forensic Recorder	Forensic recording + ERC-8004 reputation	isolator-verigate-init...	0

HOW VERIFICATION WORKS

Verigate uses Ed25519 digital signatures (not HMAC) - verification requires only the public key. No shared secrets, no trust in the operator or Circle.

Step 1: RFC 8785 JCS Canonicalization

The receipt body is serialized using JSON Canonicalization Scheme (RFC 8785) - deterministic key ordering and encoding. This ensures the exact same bytes are produced regardless of JSON library or language.

Step 2: SHA-256 Hash

Compute SHA-256 of the canonical bytes. This becomes the receipt_hash (prefixed with 'sha256:'). Any modification to any field produces a completely different hash.

Step 3: Ed25519 Signature Verification

The gateway signs the canonical bytes with its Ed25519 private key. Anyone with the public key can verify the signature. Ed25519 is deterministic - the same message always produces the same signature.

Step 4: Hash Chain Verification

Each receipt includes prev_receipt (the hash of the previous receipt). This creates an append-only chain. Inserting, deleting, or reordering receipts breaks the chain.

Step 5: Merkle Root + Anchor

All receipt hashes are combined into an RFC 6962 Merkle tree. The root is signed by the Circle Agent Wallet and can be anchored on-chain for immutability.

OFFLINE VERIFICATION (no trust required)

Third-party arbiters can verify the entire receipt chain offline using only Python and the exported JSON file. No network access, no credentials, no trust in Verigate or Circle.

```
# Step 1: Export the chain from the dashboard
curl http://verigate-dashboard-1031148889398.us-central1.run.app/api/export > chain-export.json

# Step 2: Verify offline (requires only Python + cryptography)
python -m circle.dispute verify chain-export.json

# Step 3: Generate a dispute resolution PDF
python -m circle.dispute report chain-export.json \
--output dispute-report.pdf
```

Manual Python Verification:

```
# Manual verification with Python:
import json, hashlib, base64
from cryptography.hazmat.primitives.asymmetric.ed25519 import Ed25519PublicKey

# Load export
with open("chain-export.json") as f:
    data = json.load(f)

# Get public key (Ed25519 JWK)
jwk = data["public_key_jwk"]
x_bytes = base64.urlsafe_b64decode(jwk["x"] + "==")
pub_key = Ed25519PublicKey.from_public_bytes(x_bytes)

# For each receipt:
for env in data["receipt_chain"]:
    body_bytes = json.dumps(env["body"],
        sort_keys=True, separators=(",",":")).encode()
    computed = "sha256:" + hashlib.sha256(body_bytes).hexdigest()
    assert computed == env["receipt_hash"], "TAMPERED!"
    sig = base64.urlsafe_b64decode(env["sig"]["value"] + "==")
    pub_key.verify(sig, body_bytes)  # raises if invalid
```

TAMPER EVIDENCE

Any modification to any receipt field - decision, amount, payee, policy hash, or timestamp - changes the canonical JSON, producing a completely different SHA-256 hash and invalidating the Ed25519 signature. The hash chain ensures no receipt can be inserted, deleted, or reordered without detection. The Merkle root provides batch-level integrity, and the on-chain anchor makes the proof externally immutable.

RELATIONSHIP TO CIRCLE

Circle protects the wallet. Verigate protects the operator. Circle's Action Gate + MPC co-signer enforces spending limits at the cryptographic layer. Verigate produces the cryptographic proof that every decision was made correctly - independently verifiable by third parties without trusting the operator or Circle. Two systems, zero overlap.

CERTIFICATION: This report is machine-generated by Verigate and certifies that the cryptographic verification described herein was performed at the stated time. The verification result is mathematically deterministic - identical inputs always produce identical outputs. Any party with the public key can independently replicate all verification steps without Verigate credentials.

REGULATORY ALIGNMENT: Receipt chain + signed audit reports satisfy evidence requirements under EU AI Act (Art 12-13), DORA Article 11, NIST AI RMF, and HIPAA §164.312(b). On-chain Merkle anchoring satisfies SEC Rule 17a-4(f) WORM requirements.