VERIFIED - All cryptographic checks passed

The Ed25519 receipt chain, hash integrity, Merkle root, x401 identity binding, and wallet-signed anchor have all been verified. This receipt chain has not been tampered with.

VERIFICATION CHECKS

Check	Result
Ed25519 Signatures	✓ PASS
Hash Chain Integrity	✓ PASS
Merkle Root (RFC 6962)	✓ PASS
x401 Identity Binding	✓ PASS
Wallet-Signed Anchor	✓ PASS
Overall Verdict	✓ PASS

RECEIPT CHAIN SUMMARY

Field	Value
Total Receipts	2
Payments Approved	1
Payments Denied	1
x401 Identity Bindings	1
Settlement Bindings	1
Signed Artifacts	8
Active Agents	6

SETTLEMENT BINDINGS

Each approved payment receipt references the on-chain USDC settlement transaction. Neither receipt nor settlement can exist without the other (Recibo binding).

Seq	Receipt Hash	Tx Hash	Amount	Chain
1	sha256:6d5390a7b75495cd26582e493...	0x0753f12385d3ec37f96f8e...	0.01 USDC	BASE-SEPOLIA

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-live-demo-ebb4cb...	1

Gateway	Deterministic policy eval + signed receipts	gateway-live-demo-df3e44f6...	2
Auditor	EU AI Act / NIST / DORA compliance proof	auditor-live-demo-e4dfc21a...	4
Investigator	Forensic evidence + severity classification	investigator-live-demo-d2cb9...	1
Recommender	Circle policy recommendations	recommender-live-demo-b02439...	1
Forensic Recorder	Forensic recording + ERC-8004 reputation	gateway-live-demo-df3e44f6...	1

SIGNED ARTIFACTS

Every artifact is independently signed by its producing agent. The SHA-256 hash can be recomputed from the canonical body.

Agent	Type	Artifact Hash (SHA-256)
coordinator	service discovery	`sha256:70ab232543c03484a3fa9da0b0afeda6ebc381ee17d96b5ab1453e649d7437ab`
auditor	audit report	`sha256:fd0b3ce045b59e0e64075ce989b2c37fdfa05c0535618cd3061016ffaae689b0`
auditor	audit report	`sha256:3fa116dc57d9102e8abff4beb51d0e58caf823d8871b623bfc52b743486e28d6`
auditor	audit report	`sha256:a07412a1bf85f6d9202ef7673c497d1e5754cd81a3d892b83b1627d36f962bd0`
auditor	compliance report	`sha256:fddc546e6c23ecabde6be17a4d440b2d6e9e4d1abf335f9155bb052e8c7fa364`
investigator	incident report	`sha256:e68cf7c2c8399e0e375a4a5e49975cc847f206b87d3d94e542f49ba8d663c124`
recommender	policy proposal	`sha256:e4ab5383901dedf559abb36745e662d929a9cc5d62bd2787f5988d4ee180a958`
forensic recorder	forensic record	`sha256:faaeeb2d74e53e533d94a17e7ebb03813c77b5c0a1b7e261d4c71a3e1d0ae350`

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.cloud/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
```

GATEWAY PUBLIC KEY (Ed25519 JWK)

This is the gateway's Ed25519 public key in JWK format. Use it to independently verify any receipt signature. The private key never leaves the gateway process.

```
{
  "kty": "OKP",
  "crv": "Ed25519",
  "kid": "gateway-live-demo-df3e44f6",
  "use": "sig",
  "alg": "EdDSA",
  "x": "lo9d60xEj2bMa9o-Yd_Fd6PUk4SL8wl3bk4VbOI9dew"
}
```

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.