To provide developers and system integrators with a complete understanding of the refugee identity management system powered by blockchain and AI, this document offers **developer-level code documentation** for the smart contracts and deployment configuration. It complements the whitepaper by explaining architecture depth, implementation details, and gas/cost implications.

✨Smart Contract Capability and Implementation Guide

🦜Can You Write Smart Contracts?

Yes. This system includes:

- **Hyperledger Fabric chaincode** for core identity and attestation logic (private network)
- **Ethereum smart contracts (Solidity)** for interoperability and public verification

---

🦬Technical Architecture Depth

⚡Blockchain Selection: Public vs Private

Feature	Hyperledger Fabric (Private)	Ethereum (Public)
Governance	Permissioned	Decentralized
Privacy	High (Private channels)	Low (Public by default)
TPS	1000+	\~15
Gas Fees	None	Paid per operation
Regulation Compliance	Easier (GDPR etc.)	Harder
Interoperability	Limited	High (via bridges)

⚡**Conclusion**: Use **Fabric** for secure, consortium-governed operations and **Ethereum** for global attestation and cross-chain interoperability.

🦅Full Smart Contract Implementations

1. Hyperledger Fabric (Private)

- Language: Node.js Chaincode
- Core Logic:
- Identity registration with encrypted metadata and biometric hash
- Biometric verification with simulated AI score
- Attestation management
- Selective service access (ZK-like simulation)
- Cross-border portability

Modules Included:

- `initLedger(ctx)` → initialize config and orgs
- `registerIdentity(ctx, id, biom, metadata, org)` → create identity
- `verifyIdentity(ctx, id, challenge, org)` → update AI score, verification level
- `addAttestation(ctx, id, type, data, issuer)`
- `grantServiceAccess(ctx, id, provider, type, required)`
- `enableCrossBorderAccess(ctx, id, destCountry, org)`
- Query methods and pagination

Helper Utilities:

- SHA-256 biometric matching + similarity calculation
- Expiry date management per attestation/service
- Event emission: registration, verification, attestation

2. Ethereum Smart Contract (Public Interop)

- Language: Solidity (v0.8.19)
- Structure:
- `RefugeeChainPublicRegistry.sol`
- `RefugeeChainInteroperability.sol`

Key Features:

- On-chain attestation with evidence hashes
- Public cross-chain record management
- Biometric verification challenges and confidence scoring
- Zero-knowledge proof simulation for access control
- Role-based access (admin, validator, attestor, oracle)
- Batch operations for attestations
- Identity revocation with `EMERGENCY_ROLE`

🐒 Gas Cost and Performance Analysis

Ethereum Gas Consumption

Operation	Gas Used	ETH Cost (@20 gwei)	USD Cost (@\$2000)
Add Attestation	180,000	0.0036 ETH	\$7.20
Verify Challenge (Oracle)	75,000	0.0015 ETH	\$3.00
Grant Service Access	120,000	0.0024 ETH	\$4.80
Emergency Revoke	85,000	0.0017 ETH	\$3.40

Batch operations reduce per-unit cost significantly.

Fabric Transaction Cost Estimate

Operation	CPU Time (s)	Memory (MB)	Notes
Register Identity	0.015	20	Encrypted + biometric index
Verify Identity	0.008	10	AI-based simulation
Add Attestation	0.012	12	SHA hash indexing
Grant Access	0.010	8	ZK hash generation

Fabric incurs no gas fees, ideal for scale.

🌳 Deployment Configuration

Ethereum Networks

```
{
  "mainnet": {
    "gasLimit": 8000000,
    "attestationFee": "0.001 ETH"
  },
  "polygon": {
    "gasLimit": 20000000,
    "attestationFee": "0.01 MATIC"
  },
  "sepolia": {
    "gasLimit": 8000000,
    "attestationFee": "0.001 ETH"
  }
}
```

Fabric Deployment

- Channel: `refugeechainchannel`
- Chaincode Name: `refugeeidentity`
- Organizations: UNHCR, IRC, UNICEF
- Consensus: RAFT
- Endorsers: `peer0`, `peer1` per org
- Certificate Authorities (CAs): Fabric CA per org

Scripts for chaincode install, approval, lifecycle and testing are provided in `deploy.sh`

🏜️ Future Enhancements

- Native ZK-SNARK integration (zkSync, Halo2)
- Off-chain compute with Chainlink OCR

- DID (Decentralized Identifiers) integration (W3C spec)
- Identity migration protocol via L2 rollups

🔐 Security and Compliance

- TLS across all peer/org communications
- Encrypted identity metadata (AES-GCM)
- GDPR-compliant selective disclosures
- Biometric hash indexing (no raw data on chain)

🧠 Developer Notes

- Smart contract tests in `Mocha + Chai` (Fabric) and `Hardhat` (Ethereum)
- Use Postman or `fabric-network` SDK for API simulations
- Ethereum interactions supported via `ethers.js`

Contact `dev@refugeechain.org` for full repo access or SDK onboarding.

🌞 Versioning

Component	Version
Hyperledger Fabric Chaincode	1.0.0
Ethereum Registry (Solidity)	1.0.0
Deployment Tooling	1.0.0
AI Verification Logic	0.9.2