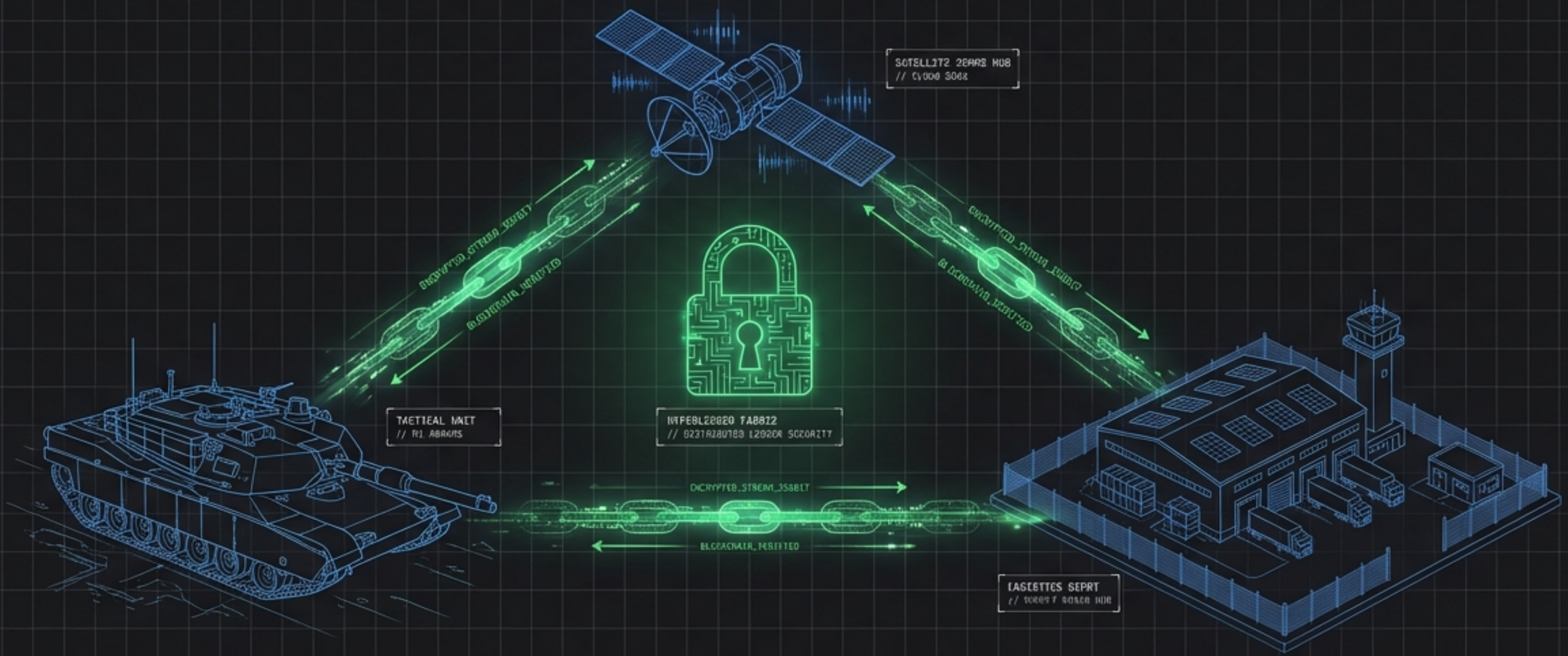
OPERATION IMMUTABLE

HYPERLEDGER FABRIC IN THE TACTICAL DOMAIN

Securing Logistics, Comms, and Data from Cloud to Edge.



UNCLASSIFIED // TECHNICAL BRIEFING

THE PROBLEM: TRUST IN A ZERO-TRUST ENVIRONMENT

CURRENT STATE: FOG OF WAR

FUTURE STATE: IMMUTABLE LEDGER

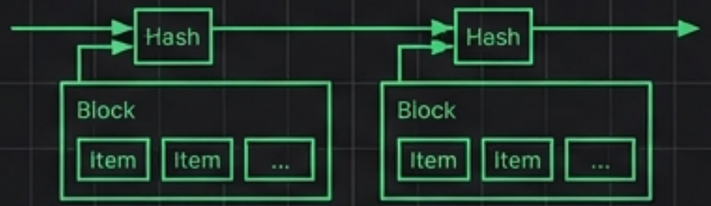
Blockchain is a tamper-proof digital logbook.
Everyone has a copy; no one can erase a page.

WHY HYPERLEDGER FABRIC? (IT IS NOT BITCOIN)

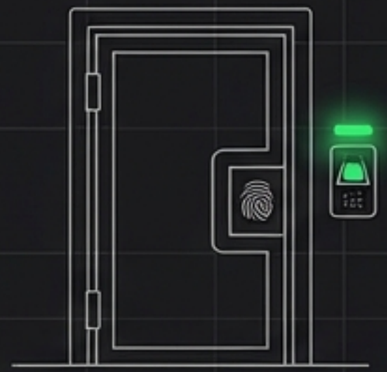
PUBLIC BLOCKCHAIN (BITCOIN)

High Energy Waste (Mining)

- Anonymous Actors
- Anyone can post
- Proof of Work = Slow



PERMISSIONED (HYPERLEDGER FABRIC)

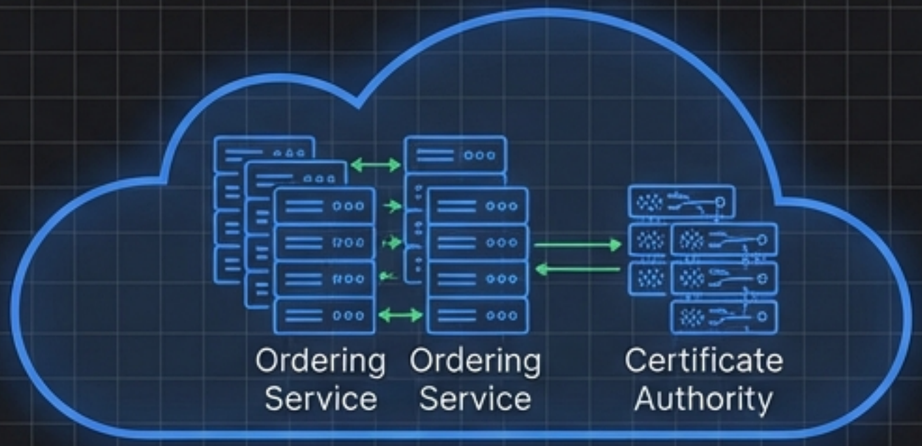


Efficient (No Mining)

- Known Identities (MSP)
- Need-to-Know Access
- Crash Fault Tolerance = Fast

INFRASTRUCTURE: CLOUD HQ TO TACTICAL EDGE

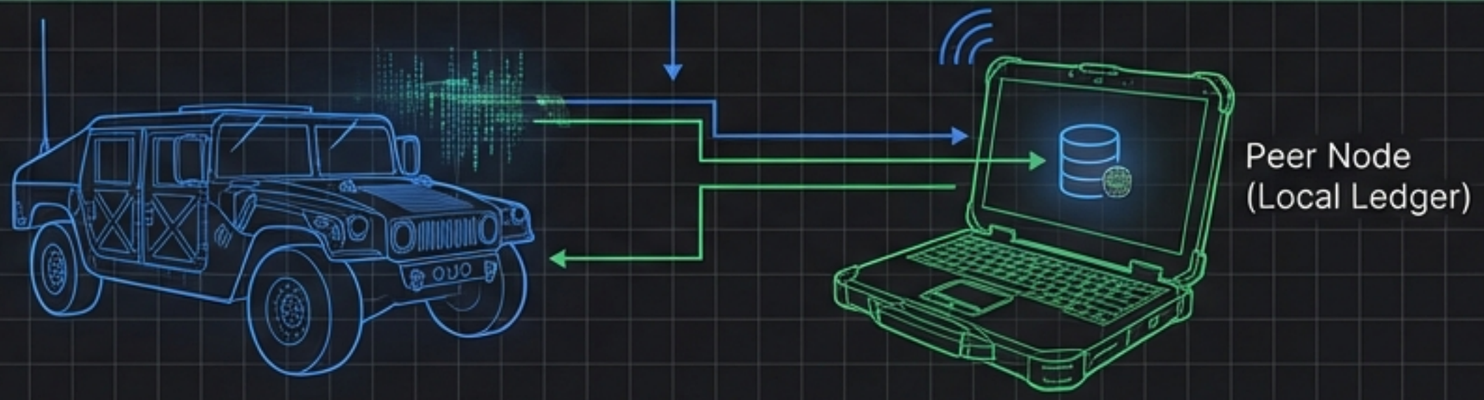
HQ / CLOUD TIER (AWS)



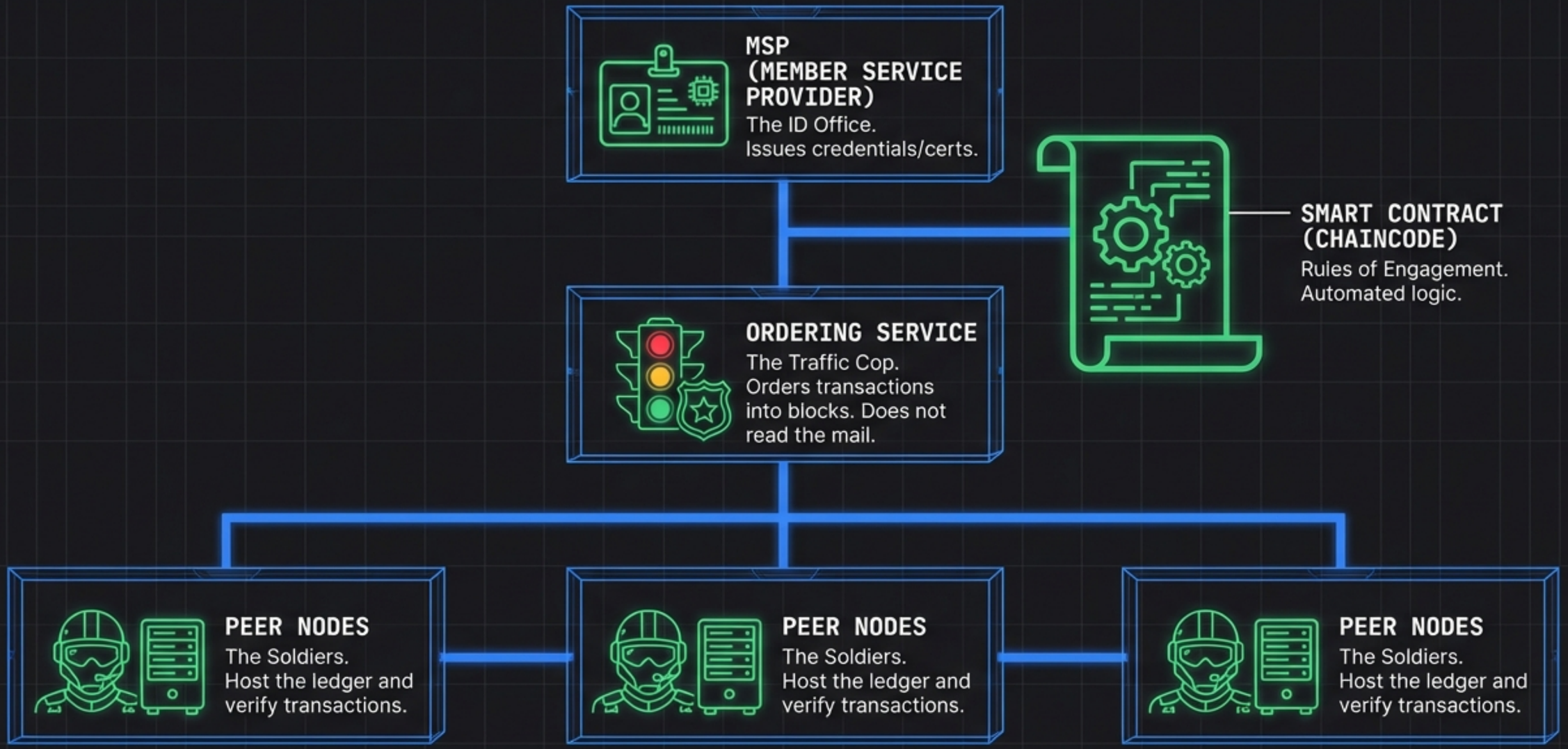
DDIL ENVIRONMENT (Disconnected/Intermittent)

Local Peers continue to transact offline.
Auto-sync via JSON/CSV on reconnect.

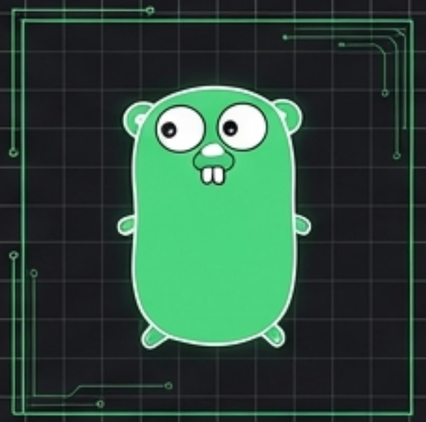
TACTICAL EDGE



THE COMPONENTS S: DIGITAL CHAIN OF COMMAND



THE LOADOUT: DEVELOPER TOOLS



Go (Golang) - Chaincode

Node.js - Client SDK

Docker - Containerization

Kubernetes - Orchestration

Git - Version Control

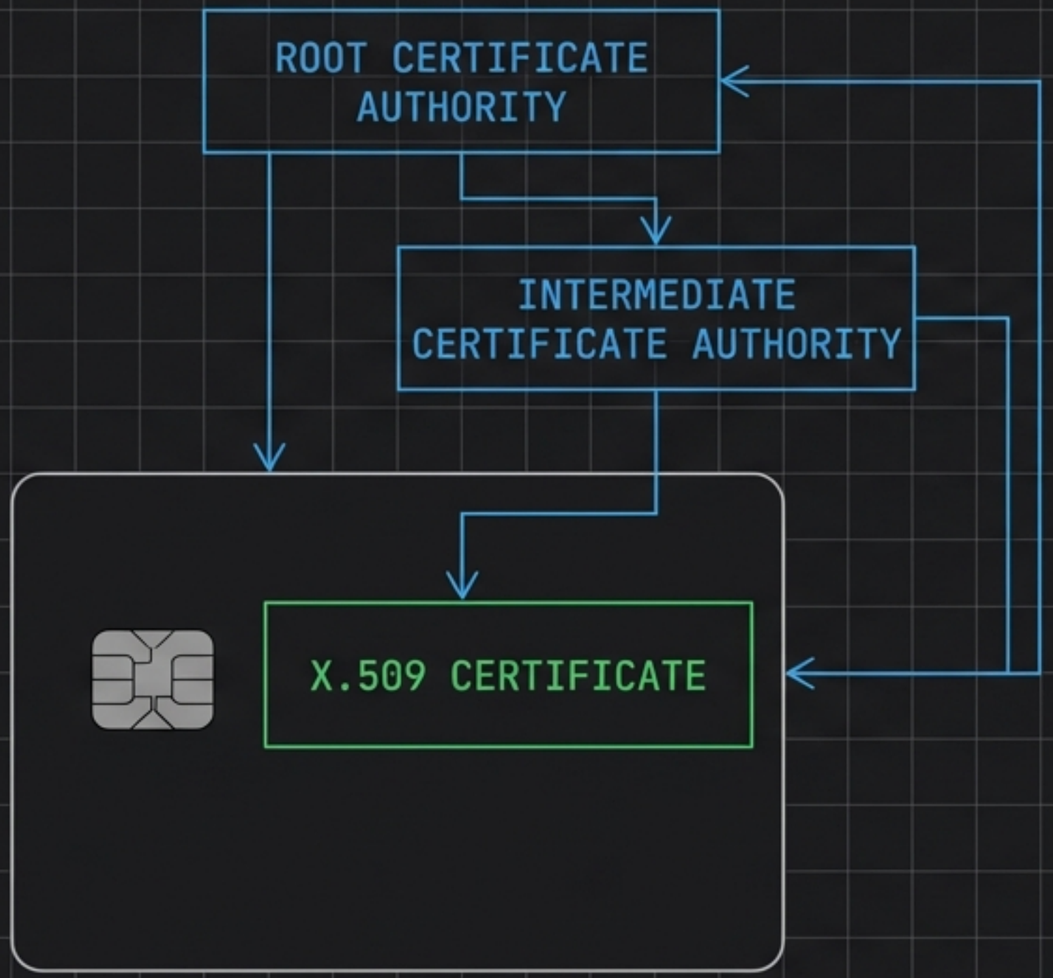
Architecture: Hyperledger Fabric (Permissioned). Not Hyperledger Besu (Ethereum).

PHASE 1: INFRASTRUCTURE AS CODE (IaC)

```yaml
version: '2'
services:
  peer0.org1.example.com:
    container_name: peer0.org1.example.com
    image: hyperledger/fabric-peer:latest
    environment:
      - CORE_PEER_ID=peer0.org1.example.com
      - CORE_PEER_LOCALMSPID=Org1MSP
    volumes:
      - /var/run/:/host/var/run/
```

Defining the Network Topology: Spinning up a Peer Node via Docker Compose.

PHASE 2: IDENTITY & ACCESS (THE DIGITAL GATE GUARD)



NODE.JS CLIENT SDK

```
// Enroll the admin user
const enrollment = await ca.enroll({
    enrollmentID: 'admin',
    enrollmentSecret: 'adminpw'
});

// Register a new soldier (user)
const secret = await ca.register({
    affiliation: 'org1.department1',
    enrollmentID: 'soldier1',
    role: 'client'
}, adminUser);
```

PHASE 3: CHAINCODE (THE SMART CONTRACT)

BUSINESS LOGIC

Define Asset: Spare Part

Fields: ID, Name, Owner, Value

Function: Initialize Ledger

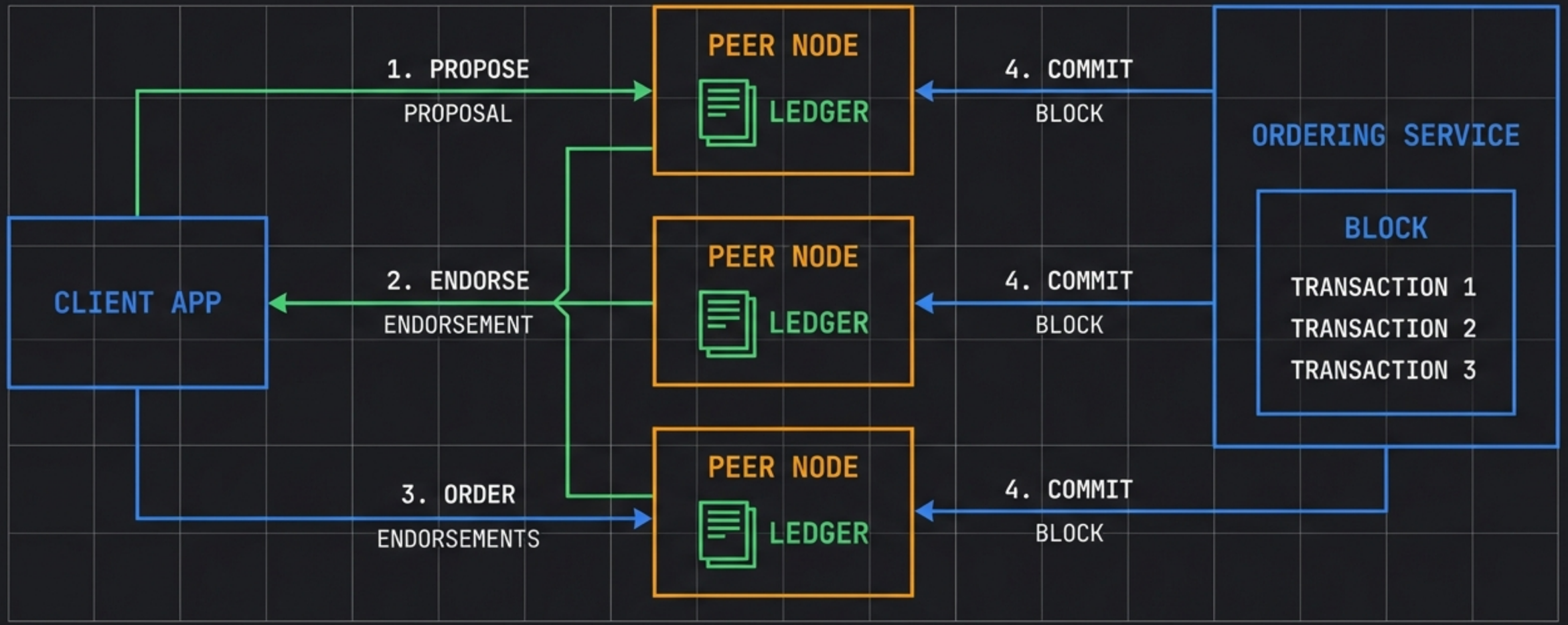
GO IMPLEMENTATION

```go
type Asset struct {
    ID             string `json:"ID\"`
    Name           string `json:"name\"`
    Owner          string `json:"oener\"`
    AppraisedValue int    `json:"appraisedValue\"
}

func (s *SmartContract) InitLedger(ctx contractapi.TransactionContextInterface) error {
    assets := []Asset{
        {ID: "PART-001", Name: "N1 Track Link", Owner: "Depot-A", AppraisedValue: 300},
        {ID: "PART-002", Name: "Drone Battery", Owner: "FOB-Bravo", AppraisedValue: 150},
    }
    // Committing assets...
}

```

PHASE 4: THE dAPP (SOLDIER INTERFACE)

```
// Connect to gateway
await gateway.connect(ccp, { wallet, identity: 'soldier1' });

// Submit transaction
await contract.submitTransaction('TransferAsset', 'PART-001','FOB-Bravo');
```

PHASE 5: THE TRANSACTION FLOW

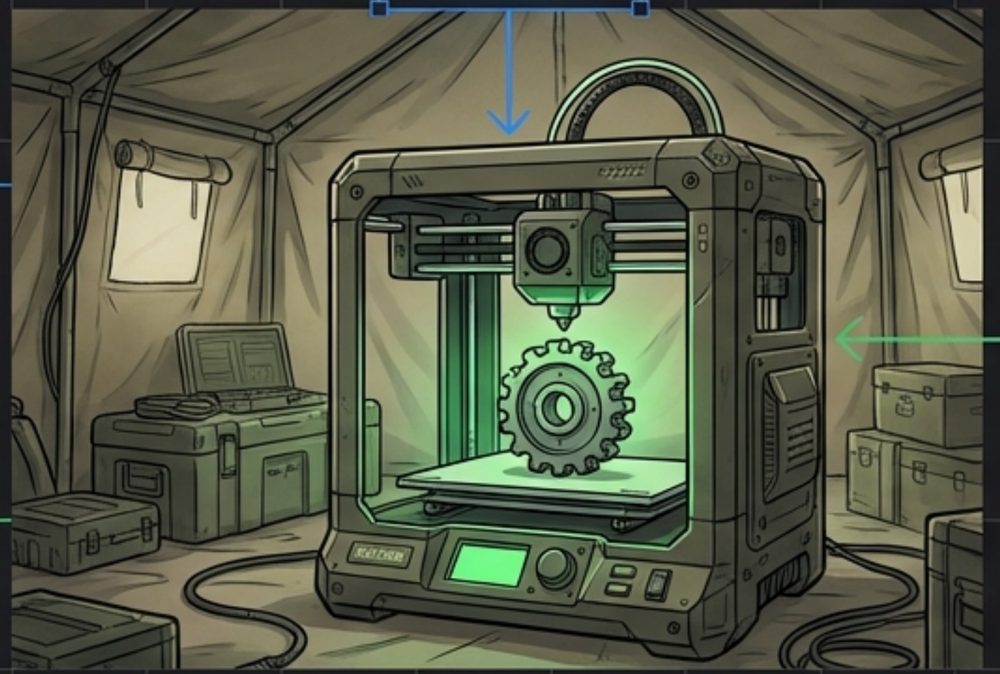


USE CASE: DIGITAL THREAD FOR LOGISTICS

PROBLEM: HACKED FILES = DEFECTIVE PARTS.

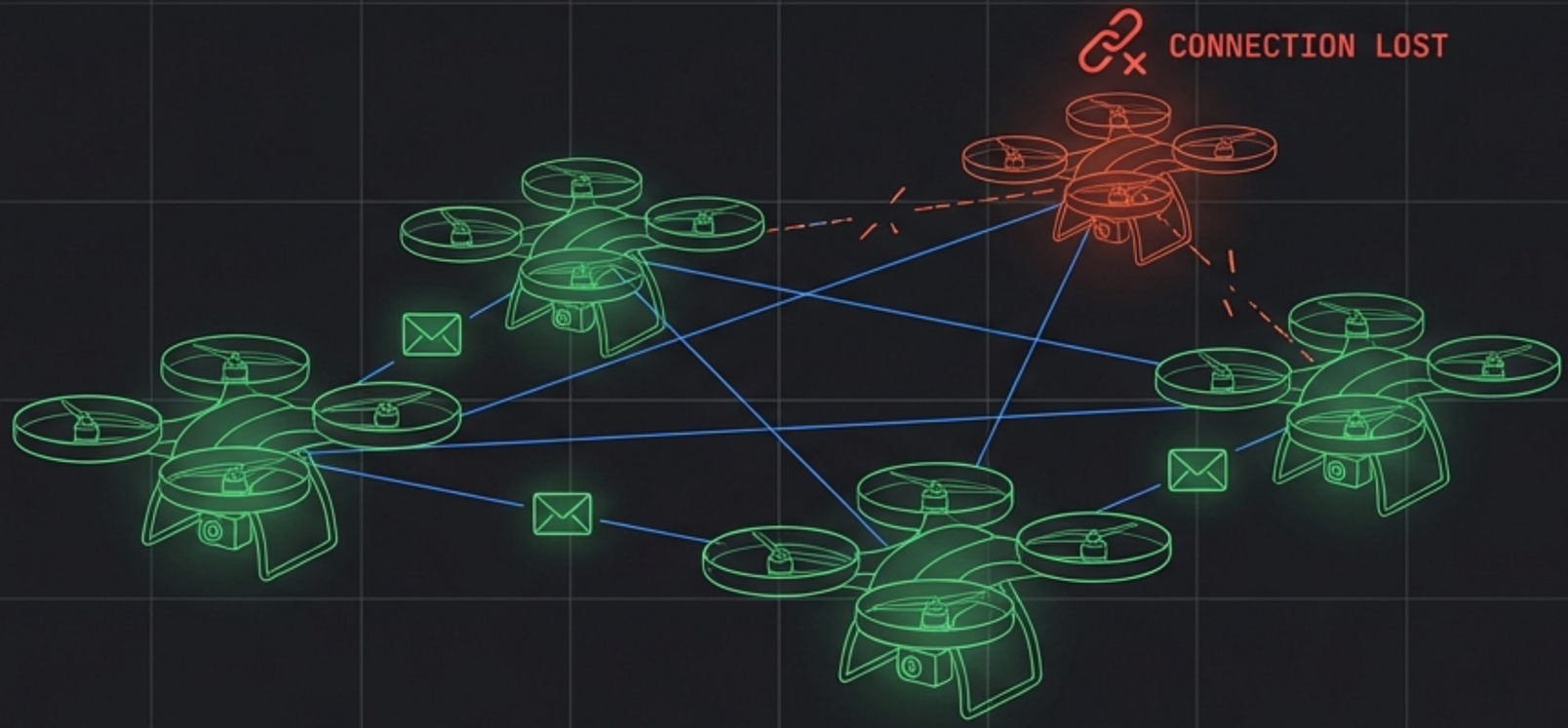
SOLUTION: IMMUTABLE PROVENANCE.

RESULT: VERIFIED INTEGRITY FROM DESIGN TO PRINT.



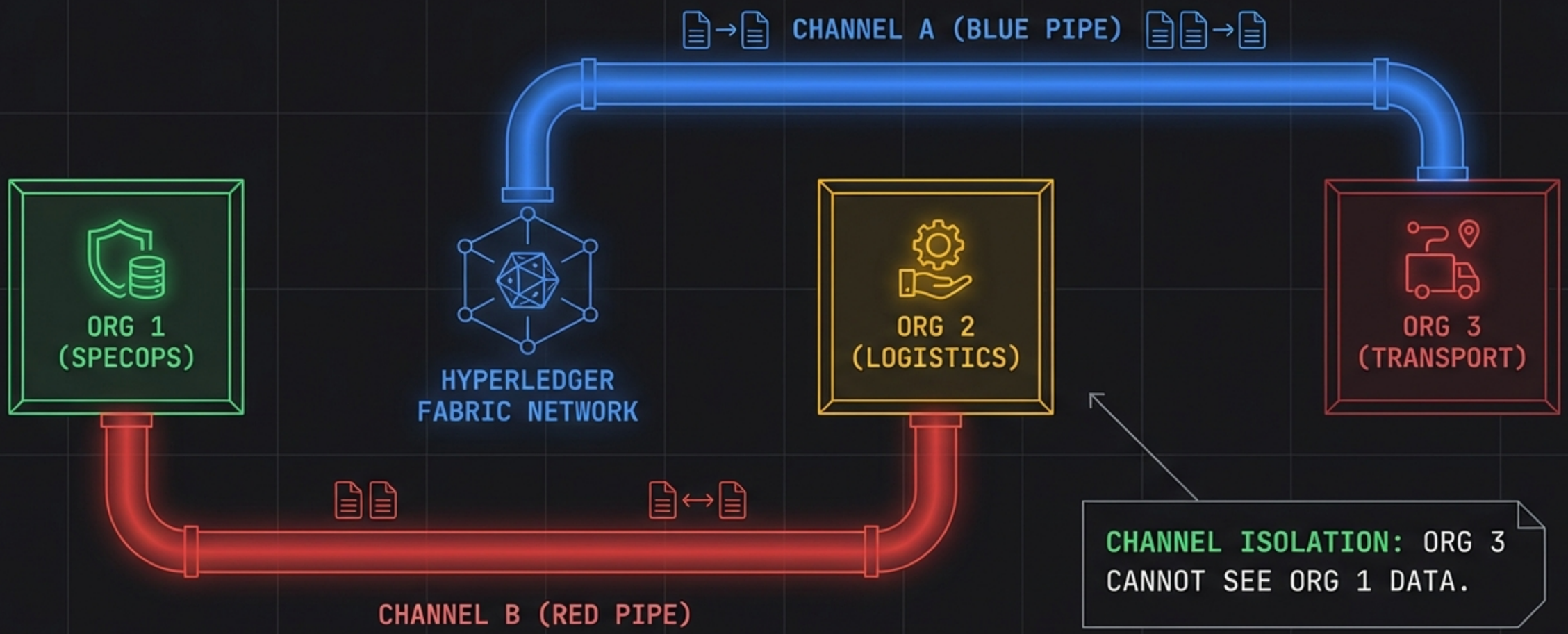
BLOCKCHAIN VERIFIED

USE CASE: SECURE DRONE SWARM LOGS



Decentralized Intel. If one node is lost, the mission data survives on the swarm ledger.

CONFIGURATION: PRIVACY VIA CHANNELS



COMMANDER'S INTENT: TRUSTED & RESILIENT

SECURITY: Tamper-proof logs prevent fraud and cyber-sabotage.

HIERARCHY: Permissioned MSP structure mirrors Chain of Command.

RESILIENCE: Operates in disconnected (DDIL) environments.

RECOMMENDATION: Initiate Pilot Program for Class IX (Repair Parts) Tracking.

"Innovation is driven by the potential to solve existing problems in new ways." — Brig. Gen. Mark T. Simerly