IRSB Protocol: Operator-Grade System Analysis

For: DevOps Engineer Generated: 2026-01-26 Version: f4cc1a5 (master)

---

1. Executive Summary

Business Purpose

IRSB (Intent Receipts & Solver Bonds) is an Ethereum-based accountability layer for intent-based transactions, complementing the ERC-7683 cross-chain intents standard. The protocol addresses a critical gap in the DeFi ecosystem: the lack of verifiable accountability for "solvers" - entities that execute user intents (swaps, bridges, complex trades) on their behalf.

The protocol is currently deployed on Sepolia testnet with a fully operational tech stack including smart contracts, TypeScript SDK, GraphQL subgraph, and Next.js dashboard. The system is pre-mainnet, having completed security audit preparation and outreach to Tier 1 audit firms. Test coverage stands at 95 tests passing across 3 core contracts (~900 SLOC).

The technology foundation is robust: Solidity 0.8.25 contracts using OpenZeppelin security primitives, Foundry for testing/deployment, The Graph for indexing, Firebase Hosting for the dashboard, and GitHub Actions for CI/CD. The architecture follows a modular design with clear separation between registration (SolverRegistry), execution proof (IntentReceiptHub), and dispute resolution (DisputeModule).

Primary operational risks include: (1) no production monitoring infrastructure yet established, (2) mainnet deployment pending security audit completion, (3) single-key deployment model (no multisig), and (4) centralized arbitrator role for subjective disputes.

Operational Status Matrix

Environment	Status	Uptime Target	Release Cadence
Production (Mainnet)	Not Deployed	99.9% (target)	Post-audit
Testnet (Sepolia)	Live	Best effort	As needed
Dashboard	Live	99%	On push to master
Subgraph	Live	99%	Manual deploy

Technology Stack

Category	Technology	Version	Purpose
Smart Contracts	Solidity	0.8.25	Core protocol logic
Contract Framework	Foundry	nightly	Build, test, deploy
Libraries	OpenZeppelin	5.x	Security primitives
SDK	TypeScript	5.3+	Client integration
Indexing	The Graph	0.0.5	Event indexing
Dashboard	Next.js	14.x	Public UI
Hosting	Firebase	-	Dashboard hosting
CI/CD	GitHub Actions	-	Automated testing

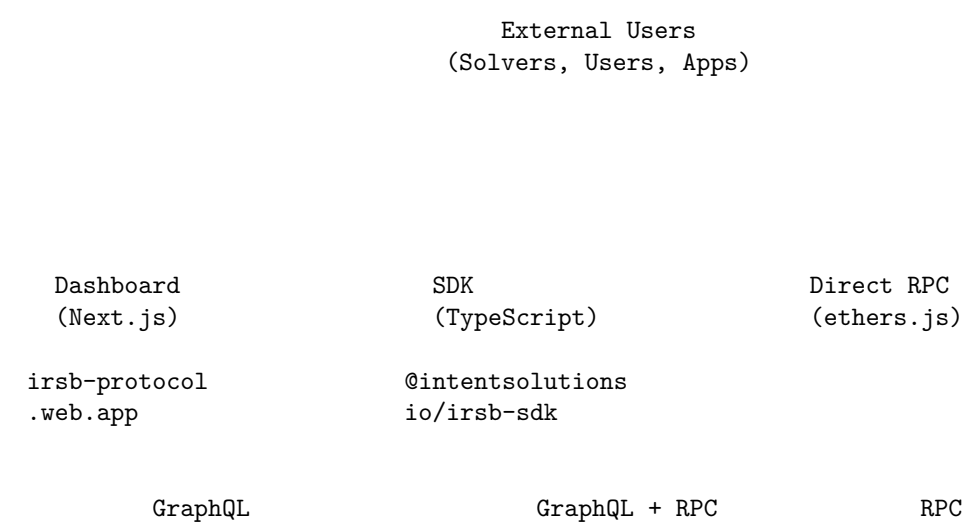
---

2. System Architecture

Technology Stack (Detailed)

Layer	Technology	Version	Purpose	Owner
Smart Contracts				
Core	Solidity	0.8.25	Protocol logic	Protocol team
Security	OpenZeppelin	5.x	ReentrancyGuard, Ownable, Pausable	External
Testing	Foundry	nightly	forge test, forge script	Protocol team
Backend				
Indexing	The Graph	0.0.5 spec	Event indexing, GraphQL API	Protocol team
ABIs	JSON	-	Contract interfaces	Auto-generated
Frontend				
Framework	Next.js	14.x	React SSR/SSG	Protocol team
Styling	Tailwind CSS	3.4.x	Utility-first CSS	Protocol team
Web3	ethers.js	6.9.x	Blockchain interaction	External
GraphQL	graphql-request	7.4.x	Subgraph queries	External
Infrastructure				
Hosting	Firebase Hosting	-	Static site hosting	Google
CI/CD	GitHub Actions	-	Automated pipelines	GitHub
RPC	Alchemy/Infura	-	Ethereum node access	External
SDK				
Language	TypeScript	5.3.x	Type-safe client	Protocol team
Bundler	tsup	8.x	ESM/CJS builds	External
Testing	Vitest	1.x	Unit tests	External

Architecture Diagram



The Graph (Subgraph)

Entities: Solver, Receipt, Dispute, ProtocolStats, DailyStats
Network: Sepolia
Studio: https://thegraph.com/studio/subgraph/irsb-protocol

Events

Ethereum (Sepolia Testnet)

SolverRegistry	IntentReceiptHub	DisputeModule
0xB6ab...ecB745	0xD66A...d977c	0x144D...B79D
• registerSolver()	• postReceipt()	• submitEvidence()
• depositBond()	• openDispute()	• escalate()
• withdrawBond()	• finalize()	• resolve()
• slash()	• resolveDetermin()	• resolveByTimeout()
• jailSolver()		

Authorization Flow:
- SolverRegistry.setAuthorizedCaller(IntentReceiptHub, true)
- SolverRegistry.setAuthorizedCaller(DisputeModule, true)
- IntentReceiptHub.setDisputeModule(DisputeModule)

Data Flow

```
1. SOLVER REGISTRATION
   Solver    SolverRegistry.registerSolver()
             SolverRegistry.depositBond() [>= 0.1 ETH]
             Status: Active

2. RECEIPT POSTING
   Solver executes intent off-chain
             IntentReceiptHub.postReceipt(signedReceipt)
             Challenge window opens (1 hour)

3. HAPPY PATH (No Dispute)
   Challenge window expires
             Anyone calls IntentReceiptHub.finalize()
             Solver reputation updated

4. DISPUTE PATH
   Challenger    IntentReceiptHub.openDispute() [+ 0.01 ETH bond]
                 Solver bond locked
                 Deterministic check OR escalation to DisputeModule
                 Resolution: slash solver OR release bond
```

3. Directory Analysis

Project Structure

```
irsb-protocol/
  000-docs/                    # Project documentation
     diagrams/                 # Educational diagrams (PNG, PDF)
     001-RL-PROP-*.md          # Proposal documents
     002-PP-PROD-*.md          # PRD
     003-AT-SPEC-*.md          # EIP specification
     ...                       # Research, status reports

  audit/                       # Security audit package
     SCOPE.md                  # Audit scope definition
     THREAT-MODEL.md           # Attack vectors, mitigations
     INVARIANTS.md             # Formal protocol invariants
     README.md                 # Quick start for auditors

  broadcast/                   # Foundry deployment artifacts
     Deploy.s.sol/             # Per-chain deployment records

  dashboard/                   # Next.js public dashboard
     src/                      # React components, pages
     public/                   # Static assets
     firebase.json             # Firebase hosting config
     package.json              # Node dependencies

  deployments/                 # Deployment records
     sepolia.json              # Contract addresses, config

  lib/                         # Git submodules (Foundry deps)
     forge-std/                # Foundry standard library
     openzeppelin-contracts/   # OpenZeppelin contracts

  outreach/                    # Partnership/auditor outreach
     auditor-outreach.md       # Audit firm tracking
     *-outreach.md             # Partner outreach templates

  script/                      # Deployment scripts
     Deploy.s.sol              # Main deployment script
     SeedTestData.s.sol        # Test data seeding
     SeedReceiptsOnly.s.sol    # Receipt seeding

  sdk/                         # TypeScript SDK
     src/                      # SDK source code
     dist/                     # Built output
     examples/                 # Usage examples
     package.json              # NPM package config

  src/                         # Smart contract source
     interfaces/               # Contract interfaces (I*.sol)
     libraries/                # Shared types, events
     adapters/                 # Protocol adapters
     SolverRegistry.sol        # Core: registration, bonds
     IntentReceiptHub.sol      # Core: receipts, disputes
```

```
    DisputeModule.sol        # Core: arbitration

subgraph/                    # The Graph subgraph
    src/                     # Event handlers (AssemblyScript)
    abis/                    # Contract ABIs
    schema.graphql           # GraphQL schema
    subgraph.yaml            # Subgraph manifest

test/                        # Foundry tests
    SolverRegistry.t.sol     # 36 tests
    IntentReceiptHub.t.sol   # 38 tests
    DisputeModule.t.sol      # 21 tests
    AcrossAdapter.t.sol      # Adapter tests

.github/workflows/           # CI/CD pipelines
    ci.yml                   # Main CI (tests, lint, build)
    deploy-dashboard.yml     # Dashboard deployment
    release-sdk.yml          # SDK publishing
    test.yml                 # Basic test workflow

.env.example                 # Environment template
foundry.toml                 # Foundry configuration
CLAUDE.md                    # AI assistant context
README.md                    # Project overview
```

Key Directories

src/ - Smart Contracts **Patterns**: Modular contract design with clear separation of concerns. Each contract inherits OpenZeppelin's `Ownable`, `ReentrancyGuard`, and `Pausable` for security. Custom modifiers for access control (`onlyAuthorized`, `onlyArbitrator`).

Entry Points: - `SolverRegistry.sol`: `registerSolver()`, `depositBond()`, `withdrawBond()` - `IntentReceiptHub.sol`: `postReceipt()`, `openDispute()`, `finalize()` - `DisputeModule.sol`: `submitEvidence()`, `escalate()`, `resolve()`

Integrations: OpenZeppelin ECDSA for signature verification, MessageHashUtils for EIP-191 compliance.

test/ - Test Suite **Framework**: Foundry forge-std **Coverage**: 95 tests across 4 test files **Gaps**: No formal verification (Certora/Echidna). Fuzz runs at 256 (default), 1000 in CI.

Test File	Tests	Coverage Focus
SolverRegistry.t.sol	36	Registration, bonds, slashing, jailing
IntentReceiptHub.t.sol	38	Receipts, disputes, finalization
DisputeModule.t.sol	21	Evidence, escalation, arbitration
AcrossAdapter.t.sol	-	Protocol adapter integration

sdk/ - TypeScript SDK **Package**: `@intentsolutionsio/irsb-sdk` (npm public) **Build**: tsup (ESM + CJS + DTS) **Dependencies**: ethers v6, graphql-request **Status**: v0.1.0, not yet published to npm

subgraph/ - The Graph **Network**: Sepolia **Entities**: Solver, Receipt, Dispute, ProtocolStats, DailyStats **Start Block**: 10100000 **Deployment**: The Graph Studio (`irsb-protocol`)

4. Operational Reference

Deployment Workflows

Local Development **Prerequisites**: - Foundry (forge, anvil, cast): `curl -L https://foundry.paradigm.xyz | bash && foundryup` - Node.js 20.x: `nvm install 20` - Git with submodule support

Setup:

```
# Clone with submodules
git clone --recursive https://github.com/intent-solutions-io/irsb-protocol
cd irsb-protocol

# Install Foundry dependencies
forge install

# Build contracts
forge build

# Run tests
forge test

# Start local Ethereum node
anvil &

# Deploy locally
forge script script/Deploy.s.sol:DeployLocal --fork-url http://localhost:8545 --broadcast
```

Verification (smoke tests):

```
# Verify contracts compiled
ls out/SolverRegistry.sol/SolverRegistry.json

# Run full test suite with gas report
forge test --gas-report

# Check contract sizes
forge build --sizes
```

Testnet Deployment (Sepolia) **Pre-flight checklist**: - [ ] `.env` file configured with `PRIVATE_KEY`, `SEPOLIA_RPC_URL`, `ETHERSCAN_API_KEY` - [ ] Deployer wallet funded with >= 0.1 ETH - [ ] All tests passing (`forge test`) - [ ] Code formatted (`forge fmt --check`)

Execution:

```
# Load environment
source .env

# Deploy to Sepolia
forge script script/Deploy.s.sol:DeploySepolia \
  --rpc-url $SEPOLIA_RPC_URL \
  --broadcast \
  -vvvv

# Verify on Etherscan (for each contract)
forge verify-contract <ADDRESS> SolverRegistry --chain sepolia
forge verify-contract <ADDRESS> IntentReceiptHub --chain sepolia --constructor-args $(cast abi-encode "
forge verify-contract <ADDRESS> DisputeModule --chain sepolia --constructor-args $(cast abi-encode "cons
```

Post-deployment: 1. Update `deployments/sepolia.json` with new addresses 2. Update `subgraph/subgraph.yaml` with new addresses 3. Rebuild and redeploy subgraph 4. Verify authorization configuration on-chain

Production Deployment (Mainnet) **BLOCKED**: Awaiting security audit completion.

Pre-flight checklist: - [ ] Security audit complete with no critical/high findings - [ ] All audit findings remediated - [ ] Multisig wallet configured for owner role - [ ] Arbitrator address designated (multisig or trusted entity) - [ ] Emergency procedures documented - [ ] Monitoring infrastructure deployed

Rollback Protocol:

Contracts are immutable. Rollback options: 1. **Pause**: Call `pause()` on all contracts (owner only) 2. **Migrate**: Deploy new contracts, migrate state via off-chain snapshot + new deployment 3. **Abandon**: If catastrophic, abandon old contracts (funds may be locked)

Dashboard Deployment

Automatic: Deploys on push to `master` when `dashboard/**` changes.

Manual:

```
cd dashboard
npm ci
npm run build
firebase deploy --only hosting
```

Firebase Project: `irsb-protocol` **URL**: https://irsb-protocol.web.app

Subgraph Deployment

```
cd subgraph
npm ci
npm run codegen
npm run build
npm run deploy:studio  # Requires graph auth
```

Studio URL: https://thegraph.com/studio/subgraph/irsb-protocol

SDK Publishing

```
cd sdk
npm ci
npm run build
npm publish --access public
```

Workflow: `.github/workflows/release-sdk.yml` (manual trigger)

Monitoring & Alerting

Current State: No production monitoring infrastructure.

Recommended Setup:

Component	Tool	Purpose
Contract Events	The Graph	Index all events
Uptime	UptimeRobot	Dashboard availability
RPC Health	Alchemy Dashboard	Node reliability

Component	Tool	Purpose
Gas Prices	Blocknative	Transaction optimization
Security	Forta	Real-time threat detection

Key Metrics to Track: - Total value bonded (TVB) - Active solver count - Receipt finalization rate - Dispute rate (disputes/receipts) - Slashing events - Gas costs per operation

SLIs/SLOs (Proposed): | SLI | Target SLO | |—|————| | Dashboard availability | 99.9% | | Subgraph sync latency | < 30 blocks | | Receipt finalization success | > 99% | | Dispute resolution time | < 7 days |

Incident Response

Severity	Definition	Response Time	Playbook
P0	Contract exploit, fund loss	Immediate	Pause all contracts
P1	Subgraph down, dashboard down	15 min	Restart services
P2	Slow sync, UI bugs	1 hour	Standard debugging
P3	Minor issues	24 hours	Queue for next sprint

Emergency Contacts: - Protocol Lead: jeremy@intentsolutions.io - GitHub Issues: github.com/intent-solutions-io/irsb-protocol/issues

---

5. Security & Access

IAM

Role	Purpose	Permissions	MFA
Contract Owner	Admin operations	`pause()`, `setAuthorizedCaller()`, `setChallengeWindow()`	Hardware wallet
Arbitrator	Dispute resolution	`resolve()` in DisputeModule	Hardware wallet
Deployer	Contract deployment	Deploy scripts	Hardware wallet
GitHub Admin	Repository access	Push to master, secrets management	Required
Firebase Admin	Dashboard deployment	`firebase deploy`	Required

Smart Contract Access Control

```
SolverRegistry
  onlyOwner
      setAuthorizedCaller()
      unjailSolver()
      banSolver()
      pause() / unpause()
      updateScore()
  onlyAuthorized (IntentReceiptHub, DisputeModule)
      lockBond()
      unlockBond()
      slash()
      jailSolver()
```

```
      incrementDisputes()
  onlyOperator (solver's own key)
      initiateWithdrawal()
      withdrawBond()
      setSolverKey()

IntentReceiptHub
  onlyOwner
      setDisputeModule()
      setChallengeWindow()
      setSolverRegistry()
      setChallengerBondMin()
      sweepForfeitedBonds()
      pause() / unpause()
  onlyDisputeModule
      resolveEscalatedDispute()

DisputeModule
  onlyOwner
      setArbitrator()
      setArbitrationFee()
      setTreasury()
      setContracts()
      withdrawFees()
  onlyArbitrator
      resolve()
```

Secrets Management

Secret	Storage	Rotation Policy
Deployer Private Key	Hardware wallet	Never (use new wallet per deployment)
Arbitrator Private Key	Hardware wallet	As needed
`ETHERSCAN_API_KEY`	GitHub Secrets	Annually
`FIREBASE_SERVICE_ACCOUNT`	GitHub Secrets	Annually
`ALCHEMY_API_KEY`	GitHub Secrets, `.env`	On compromise

Break-glass Procedure: 1. If owner key compromised: Deploy new contracts, announce migration 2. If arbitrator key compromised: Owner calls `setArbitrator()` with new address 3. If GitHub secrets compromised: Rotate all secrets, revoke old tokens

---

6. Cost & Performance

Estimated Monthly Costs

Service	Testnet	Mainnet (Projected)
Alchemy RPC	Free tier	$49/mo (Growth)
The Graph	Free (Studio)	$50-200/mo (queries)
Firebase Hosting	Free tier	Free tier
GitHub Actions	Free tier	Free tier
Etherscan API	Free tier	Free tier

Service	Testnet	Mainnet (Projected)
Total	**$0**	**$100-250/mo**

Gas Costs (Sepolia Baseline)

Operation	Gas	@ 30 gwei	@ 100 gwei
registerSolver	~150k	0.0045 ETH	0.015 ETH
depositBond	~50k	0.0015 ETH	0.005 ETH
postReceipt	~200k	0.006 ETH	0.02 ETH
openDispute	~150k	0.0045 ETH	0.015 ETH
finalize	~80k	0.0024 ETH	0.008 ETH
withdrawBond	~100k	0.003 ETH	0.01 ETH

Performance Baseline

Metric	Current	Target
Contract deployment	~3M gas total	-
Subgraph sync	< 30 blocks	< 15 blocks
Dashboard load (LCP)	~2s	< 1.5s
SDK bundle size	~150KB	< 100KB

7. Current State Assessment

What's Working

Smart Contracts: 3 core contracts deployed on Sepolia, all 95 tests passing, verified on Etherscan.

CI/CD Pipeline: Automated testing on PR, dashboard auto-deploy, SDK build verification.

Documentation: Comprehensive audit package (SCOPE.md, THREAT-MODEL.md, INVARIANTS.md), educational diagrams, EIP spec draft.

Subgraph: Deployed to The Graph Studio, indexing all protocol events.

Dashboard: Live at irsb-protocol.web.app, shows solver rankings and protocol stats.

SDK: Built and typed, ready for npm publish.

Security Prep: 10 attack vectors documented, 20+ formal invariants defined, Slither integration (non-blocking).

Areas Needing Attention

No Production Monitoring: No alerting, no dashboards, no on-call rotation.

Single-Key Deployment: Owner is single EOA, not multisig.

Centralized Arbitrator: Single address can resolve disputes with finality.

No Timelock: Parameter changes take effect immediately.

SDK Not Published: `@intentsolutionsio/irsb-sdk` not on npm yet.

Slither Warnings: Security scan runs but findings not triaged (continues on error).

No Formal Verification: Invariants documented but not verified with Certora/Echidna.

Immediate Priorities

Priority	Issue	Impact	Owner	Status
High	Security audit completion	Blocks mainnet	Protocol Lead	Awaiting auditor response
High	Multisig for owner	Reduces key risk	DevOps	Not started
Medium	Monitoring setup	Enables incident response	DevOps	Not started
Medium	SDK npm publish	Enables integrations	Protocol Lead	Ready
Low	Formal verification	Higher assurance	Security	Future

---

8. Quick Reference

Command Map

Capability	Command	Notes
Build contracts	`forge build`	Outputs to `out/`
Run all tests	`forge test`	95 tests
Run tests verbose	`forge test -vvvv`	Full traces
Gas report	`forge test --gas-report`	Per-function gas
Format code	`forge fmt`	Apply formatting
Check format	`forge fmt --check`	CI uses this
Coverage report	`forge coverage --report summary`	Branch coverage
Local node	`anvil`	Runs on :8545
Deploy local	`forge script script/Deploy.s.sol:DeployLocal --fork-url http://localhost:8545 --broadcast`	Uses Anvil key
Deploy Sepolia	`source .env && forge script script/Deploy.s.sol:DeploySepolia --rpc-url $SEPOLIA_RPC_URL --broadcast -vvvv`	Requires .env
Verify contract	`forge verify-contract <ADDR> <CONTRACT> --chain sepolia`	Requires API key
Build dashboard	`cd dashboard && npm run build`	Next.js static export
Deploy dashboard	`cd dashboard && firebase deploy`	Requires auth
Build SDK	`cd sdk && npm run build`	ESM + CJS + types
Build subgraph	`cd subgraph && npm run codegen && npm run build`	AssemblyScript
Deploy subgraph	`cd subgraph && npm run deploy:studio`	Requires graph auth

Critical URLs

Resource	URL
Dashboard (Live)	https://irsb-protocol.web.app
GitHub Repo	https://github.com/intent-solutions-io/irsb-protocol
Subgraph Studio	https://thegraph.com/studio/subgraph/irsb-protocol
SolverRegistry (Sepolia)	https://sepolia.etherscan.io/address/0xB6ab964832
IntentReceiptHub (Sepolia)	https://sepolia.etherscan.io/address/0xD66A1e880A
DisputeModule (Sepolia)	https://sepolia.etherscan.io/address/0x144DfEcB57
CI Pipeline	https://github.com/intent-solutions-io/irsb-protocol/actions
ERC-7683 Spec	https://eips.ethereum.org/EIPS/eip-7683

Contract Addresses (Sepolia)


```
{
  "SolverRegistry": "0xB6ab964832808E49635fF82D1996D6a888ecB745",
  "IntentReceiptHub": "0xD66A1e880AA3939CA066a9EA1dD37ad3d01D977c",
  "DisputeModule": "0x144DfEcB57B08471e2A75E78fc0d2A74A89DB79D",
  "Deployer": "0x83A5F432f02B1503765bB61a9B358942d87c9dc0"
}
```


Protocol Constants

Constant	Value	Contract
MINIMUM_BOND	0.1 ETH	SolverRegistry
WITHDRAWAL_COOLDOWN	7 days	SolverRegistry
MAX_JAILS	3	SolverRegistry
DECAY_HALF_LIFE	30 days	SolverRegistry
CHALLENGE_WINDOW	1 hour	IntentReceiptHub
CHALLENGER_BOND_BPS	1000 (10%)	IntentReceiptHub
EVIDENCE_WINDOW	24 hours	DisputeModule
ARBITRATION_TIMEOUT	7 days	DisputeModule
DEFAULT_ARBITRATION_FEE	0.01 ETH	DisputeModule

First-Week Checklist

- ☐ Repository access granted (GitHub)
- ☐ Cloud access granted (Firebase, Alchemy)
- ☐ Local environment running (`forge build && forge test`)
- ☐ Completed local deployment (`DeployLocal`)
- ☐ Reviewed all files in `audit/` directory
- ☐ Reviewed CI/CD workflows in `.github/workflows/`
- ☐ Joined team communication channels
- ☐ Reviewed `deployments/sepolia.json`
- ☐ Understood on-chain authorization model
- ☐ Read CLAUDE.md and README.md

9. Recommendations Roadmap

Week 1 – Stabilization

Goals: 1. Triage Slither findings → Create issues for any real vulnerabilities 2. Set up basic monitoring (UptimeRobot for dashboard) 3. Document current deployment process end-to-end 4. Publish SDK to npm (`npm publish --access public`)

Measurable Outcomes: - Slither output reviewed, 0 critical findings - Dashboard uptime monitoring active - DEPLOYMENT.md runbook created - SDK available at npmjs.com/@intentsolutionsio/irsb-sdk

Month 1 – Foundation

Goals: 1. Deploy Gnosis Safe multisig for owner role 2. Set up Forta bot for contract monitoring 3. Create testnet incident response playbook 4. Add fuzz test coverage (increase runs to 10,000 in CI) 5. Complete SDK documentation and examples

Measurable Outcomes: - Owner key migrated to 2/3 multisig - Real-time alerts for high-value events (slashing, large deposits) - Documented playbook for P0/P1 incidents - Fuzz runs at 10k in CI profile - SDK README with 3+ integration examples

Quarter 1 – Strategic

Goals: 1. Complete security audit with Tier 1 firm 2. Implement audit findings (critical/high) 3. Deploy to mainnet with multisig ownership 4. Set up comprehensive monitoring (Grafana + Prometheus) 5. Establish on-call rotation

Measurable Outcomes: - Audit report received with no unresolved critical/high findings - Mainnet contracts deployed and verified - 24/7 monitoring with PagerDuty integration - On-call rotation documented and tested - Post-launch retrospective completed

---

Appendices

A. Glossary

Term	Definition
Intent	User's desired outcome (e.g., swap 1 ETH for USDC)
Solver	Entity that executes intents on behalf of users
Receipt	Cryptographic proof of intent execution
Bond	ETH collateral staked by solvers
Slashing	Penalty where solver loses portion of bond
Challenge	Dispute opened against a receipt
IntentScore	On-chain reputation metric for solvers
Arbitrator	Trusted party that resolves subjective disputes
Finalization	Receipt confirmed valid after challenge window

B. Reference Links

- ERC-7683 Cross-Chain Intents
- Foundry Book
- The Graph Documentation
- OpenZeppelin Contracts
- Firebase Hosting

C. Troubleshooting Playbooks

Tests Failing Locally

```
# Clean build artifacts
forge clean

# Reinstall dependencies
rm -rf lib/
forge install

# Rebuild and test
forge build
forge test -vvvv
```

Subgraph Not Syncing

1. Check start block in `subgraph.yaml` (should be before first event)
2. Verify contract addresses match deployment
3. Check The Graph Studio logs for errors
4. Redeploy: `npm run deploy:studio`

Dashboard Deploy Failing

1. Check Firebase authentication: `firebase login`
2. Verify project ID in `.firebaserc`
3. Check build output: `npm run build` should produce `out/` directory
4. Manual deploy: `firebase deploy --only hosting`

Contract Verification Failing

```
# Get constructor arguments
cast abi-encode "constructor(address)" 0xB6ab964832808E49635fF82D1996D6a888ecB745

# Verify with args
forge verify-contract <ADDRESS> IntentReceiptHub \
  --chain sepolia \
  --constructor-args <ENCODED_ARGS> \
  --etherscan-api-key $ETHERSCAN_API_KEY
```

D. Open Questions

1. **Mainnet Arbitrator**: Who will serve as arbitrator? Single entity, multisig, or DAO?
2. **Timelock**: Should parameter changes have a delay? What duration?
3. **Upgradeability**: Will future versions require migration or proxy patterns?
4. **Multi-chain**: When will protocol expand beyond Ethereum L1?
5. **Insurance**: Should the protocol integrate with DeFi insurance (Nexus, UnoRe)?

Document generated by Claude Code. Last updated: 2026-01-26.