World Interview Preparation Guide

Detection & Response Internship - Verifiable Compute

Your Projects

Project 1: zk-anomaly-detector

Repository: https://github.com/OE-GOD/zk-anomaly-detector

A zero-knowledge anomaly detection system built on RISC Zero zkVM that analyzes World ID verification data while preserving privacy.

Key Features:

- Z-score based statistical anomaly detection
- Runs entirely inside zkVM (RISC Zero)
- Outputs only threat alerts, never raw data

Detection Capabilities:

Attack Type	Method
Sybil Attacks	Device ID frequency analysis
Orb Velocity Anomalies	Geographic + temporal clustering
Suspicious Clustering	Location-based pattern detection
Amount Outliers	Statistical deviation analysis

Project 2: world-zk-compute

Repository: https://github.com/OE-GOD/world-zk-compute

A Bonsol-style verifiable computation marketplace deployed on Ethereum Sepolia.

Deployed Contracts:

Contract	Address
MockRiscZeroVerifier	`0x0D194f172a3a50e0E293d0d8f21774b1a222362E`
ProgramRegistry	`0x7F9EFc73E50a4f6ec6Ab7B464f6556a89fDeD3ac`
ExecutionEngine	`0x9CFd1CF0e263420e010013373Ec4008d341a483e`

Architecture:

```
User: requestExecution() + bounty
         |
         v
Prover: claimExecution() - locks job
         |
         v
Prover: runs zkVM off-chain
         |
         v
Prover: submitProof(seal, journal)
         |
         v
Contract: verifies proof, pays prover
```

Key Innovation - Tip Decay:

```
effectiveTip = maxTip - (elapsed * maxTip / timeout)
```

Early provers earn higher rewards, incentivizing fast execution.

Technical Concepts

Why Zero-Knowledge for Detection?

Traditional Approach	ZK Approach
Send raw data to server	Prover runs analysis locally
Server analyzes data	Generates cryptographic proof
Trust the server	Verifier checks proof
Data exposed	Zero data exposure

Result: Same detection accuracy, complete privacy preservation.

RISC Zero Architecture

```
┌─────────────────────────────────────────────────┐
│                    HOST PROGRAM                  │
│  - Provides inputs to guest                     │
│  - Runs the prover                              │
│  - Extracts seal (proof) and journal (output)  │
└─────────────────────────────────────────────────┘
                      │
                      │
                      v
┌─────────────────────────────────────────────────┐
│                   GUEST PROGRAM                  │
│  - Runs inside zkVM                             │
│  - Executes detection logic                     │
│  - Commits results to journal                   │
│  - Execution is proven                          │
└─────────────────────────────────────────────────┘
                      │
                      v
┌─────────────────────────────────────────────────┐
│                     OUTPUTS                      │
│  - Seal: The ZK proof (verifiable on-chain)    │
│  - Journal: Public outputs (threat alerts)      │
└─────────────────────────────────────────────────┘
```

---

zkVMs vs Traditional ZK Circuits

Aspect	ZK Circuits	zkVMs
Language	DSL (Circom, Noir)	Standard Rust
Flexibility	Low - rewrite for changes	High - modify code easily
Learning Curve	Steep	Moderate
Proof Size	Smaller	Larger
Use Case	Fixed computations	Evolving algorithms

Why zkVMs for Detection Engineering: Detection algorithms evolve constantly. zkVMs allow writing normal Rust code and proving its execution without redesigning constraint systems.

World Company Knowledge

What is World?

- **Founded by:** Sam Altman, Alex Blania
- **Mission:** Proof-of-personhood at global scale
- **Technology:** Iris biometrics via the Orb device
- **World ID:** Privacy-preserving identity credential
- **World Chain:** L2 on OP Stack, prioritizes verified humans

Detection & Response Team Focus

1. **Sybil Resistance** - Preventing one person from creating multiple IDs
2. **Orb Fraud Detection** - Identifying fake irises, printed images, masks
3. **Operator Abuse** - Detecting compromised or malicious Orb operators
4. **Privacy Preservation** - All detection without exposing biometric data

Why Verifiable Compute Matters to World

- Run detection on sensitive biometric data
- Prove detection was performed correctly
- Never expose raw iris scans or templates
- Decentralized provers eliminate single points of trust
- Auditability without sacrificing privacy

Interview Questions & Answers

Technical Questions

Q1: "Explain how your anomaly detector works"

> *"The detector runs inside RISC Zero's zkVM. It takes World ID verification events as private input - things like device IDs, timestamps, locations, and amounts. It computes z-scores for each metric: how*

> *many standard deviations each value is from the mean. Values beyond a threshold (typically 2-3 sigma) are flagged. The zkVM commits only the threat alerts to the public journal - high-risk device IDs, suspicious locations - never the raw data. The proof guarantees the analysis was done correctly without revealing what was analyzed."*

Q2: "Why use zkVMs instead of traditional ZK circuits?"

> *"Three reasons. First, flexibility - detection algorithms evolve constantly as attackers adapt. With circuits, every change means redesigning constraints. With zkVMs, I modify Rust code and redeploy. Second, ecosystem - I can use any Rust crate. Statistical libraries, serialization, complex data structures - they all just work. Third, developer velocity - the team can iterate on detection logic without ZK expertise. The tradeoff is larger proofs and slower generation, but for batch detection jobs running hourly or daily, that's acceptable."*

Q3: "How does your bounty system prevent front-running?"

> *"Through the claim mechanism. When a prover calls claimExecution(), they lock the job exclusively for a claim period. The transaction records their address and timestamp on-chain. If another prover tries to claim, it reverts. If the original prover doesn't submit proof before the claim expires, the job unlocks and others can try. This prevents someone from watching the mempool, seeing a proof transaction, and front-running it to steal the bounty."*

Q4: "What happens if a prover claims but never submits?"

> *"The claim has a deadline - claimDeadline in the contract. After that timestamp passes, the execution status reverts to Pending. Any prover can then claim it. The original prover loses nothing except gas costs, but they also gain nothing. The tip decay mechanism means waiting costs money - a prover who claims late and submits late earns less than one who acts quickly."*

Q5: "How would you scale to 1000 requests/second?"

> *"Several approaches. First, batch multiple detection jobs into single proofs - amortize proving cost across many inputs. Second, use proof aggregation - combine multiple proofs into one for cheaper on-chain verification. Third, move to a rollup architecture where proofs are verified in batches. Fourth, implement a prover network with geographic distribution - provers close to data sources claim faster. The current design handles the common case; scaling is about parallelization and batching."*

Behavioral Questions

Q: "Why are you interested in World specifically?"

> *"World is solving one of the hardest problems in tech - proving you're human without sacrificing privacy. The intersection of biometrics, zero-knowledge proofs, and blockchain is technically fascinating. But what really draws me is the mission impact. As AI gets better at impersonation, proof-of-personhood becomes critical infrastructure. I want to build systems that billions of people rely on, and World is positioned to do exactly that."*

Q: "Describe a technical challenge you solved recently"

> *"Building the verifiable compute system. The challenge was designing incentives that work. If bounties are fixed, provers have no urgency. If they're first-come-first-served, you get front-running. I implemented tip decay - bounties decrease linearly over time. This creates natural urgency without race conditions. Provers self-select based on their costs and speed. Fast provers take low-bounty jobs; expensive jobs wait for specialized provers. It's a simple mechanism but required thinking through game theory."*

Q: "How do you approach learning new technologies?"

> *"Build something real. For this project, I'd never used RISC Zero before. I read the docs for an hour, then started coding. I hit errors, debugged them, read more docs. Within a day I had a working zkVM program. The key is choosing a project slightly beyond your current ability - not so hard you're stuck, not so easy you're not learning. Then ship it. Deployed code teaches you what tutorials can't."*

Questions to Ask the Interviewer

1. **"What's the current architecture for detection at World? On-device, centralized, or hybrid?"**
    - Shows understanding of the design space
2. **"Are you using RISC Zero, SP1, or a custom zkVM?"**
    - Demonstrates awareness of zkVM ecosystem
3. **"What's the biggest detection challenge you're facing right now?"**
    - Opens discussion about real problems you could solve
4. **"How does the team balance detection accuracy vs. privacy guarantees?"**
    - Shows you understand the core tension
5. **"What would success look like for this internship?"**
    - Practical question about expectations

Quick Reference Card

Your Elevator Pitch

> *"I built a verifiable compute platform for privacy-preserving threat detection. It's two parts: a zkVM anomaly detector that analyzes World ID data without exposing it, and a Bonsol-style bounty system where anyone can request proven computation. I deployed it to Ethereum testnet and verified the full flow works. It's directly relevant to how World could decentralize detection while maintaining privacy."*

Key Numbers

- **4** detection algorithms (Sybil, velocity, clustering, amount)
- **3** smart contracts deployed (Verifier, Registry, Engine)
- **100%** end-to-end test pass rate
- **0** raw data exposed in proofs

Technology Stack

- **zkVM:** RISC Zero
- **Contracts:** Solidity 0.8.20, Foundry
- **Prover:** Rust
- **Network:** Ethereum Sepolia (testnet)

Links

- **zk-anomaly-detector:** https://github.com/OE-GOD/zk-anomaly-detector

- **world-zk-compute:** https://github.com/OE-GOD/world-zk-compute
- **ExecutionEngine on Etherscan:** https://sepolia.etherscan.io/address/0x9CFd1CF0e263420e010013373Ec4008d341a483e

---

Prepared for World Detection & Response Internship Interview