**DeVAA: A Decentralized and Verifiable AI Agent Marketplace**

**Thesis Master Document**

**Authors:** Youssef Amjahdi, Abdelmounaim Sadir

**Supervisor:** Professor Dr. Loui Al Sardy

**Version:** 2.0

**Last Updated:** July 21, 2024

**Part I: Foundation & Background**

***Chapter 1: Introduction***

*(Primary Authors: Youssef Amjahdi, Abdelmounaim Sadir)*

**1.1 Background: The Rise of the Agent Economy**

The field of Artificial Intelligence has reached a pivotal moment. The proliferation of powerful Large Language Models (LLMs) has transcended simple question-and-answer systems, enabling the creation of autonomous AI agents. These agents, capable of reasoning, planning, and executing multi-step tasks, represent the next paradigm in software, promising to automate complex digital processes and create entirely new service economies. However, the very autonomy and complexity that make these agents powerful also introduce unprecedented challenges.

**1.2 Problem Statement: The Trust Gap**

The adoption of autonomous agents in open, decentralized ecosystems is critically hindered by a fundamental trust gap. As their internal operations are often a "black box," a user or organization has no reliable method to verify that an agent has performed a requested computation correctly. This opacity creates significant risk in any potential marketplace, exposing participants to potential fraud, incorrect results, and wasted resources. The core challenge is the absence of a framework that can facilitate a trustless exchange of services between unknown autonomous agents and users, a problem that centralized platforms cannot solve without becoming a single point of failure and control.

**1.3 Proposed Solution: The DeVAA Framework**

This thesis proposes DeVAA, a novel, decentralized marketplace framework designed specifically to solve the trust problem in the emerging AI agent economy. Our solution is built on a public blockchain, leveraging smart contracts to create a transparent, secure, and automated ecosystem. The architecture is founded on three key pillars:

* ***Verifiable Identity****:* Each AI agent is registered as a unique Non-Fungible Token (NFT), serving as a tamper-proof identity that accumulates a public history of performance and reputation.
* ***Trustless Exchange:*** A suite of smart contracts manages the service lifecycle, most notably a trustless escrow that holds a user's payment and releases it only upon successful and verifiable completion of the task.
* ***Computational Integrity:*** The core innovation is the mandatory use of Zero-Knowledge Proofs (zk-SNARKs). An agent must generate a cryptographic proof that it correctly executed the requested computation. This proof is verified on-chain, guaranteeing the integrity of the work without the agent revealing its proprietary model or the user revealing their private input data.

**1.4 Core Research Questions**

Our research is guided by the following three questions:

***RQ1:*** Architecture & Trust: How can a blockchain-based framework, utilizing smart contracts for trustless escrow and non-fungible tokens (NFTs) for identity, be designed to establish a secure, transparent, and reputation-aware marketplace for autonomous AI agents?

***RQ2:*** Verifiable Computation: To what extent can Zero-Knowledge Proofs (specifically zk-SNARKs) be practically implemented to generate a verifiable proof of computational integrity for a defined AI task, thereby enabling trustless service exchange between users and agents?

***RQ3:*** Performance & Feasibility: What are the key performance and economic trade-offs inherent in a ZKP-verified marketplace, specifically concerning on-chain gas costs, off-chain proof generation overhead, and end-to-end transaction latency, and what are their implications for the system's practical scalability?

**1.5 Thesis Contributions**

This thesis will deliver three primary contributions to the field:

* ***A Novel Architecture:*** The complete design of a decentralized marketplace for verifiable AI agents.
* ***A Practical ZKP Implementation:*** A working Proof-of-Concept demonstrating the generation and on-chain verification of a ZKP for a specific AI agent task.
* ***A Performance Benchmark:*** A quantitative analysis of the system's performance, providing a crucial, data-driven blueprint for future verifiable AI ecosystems.

**1.6 Structure of the Thesis**

This thesis is organized into three parts.

*Part I* introduces the problem domain and reviews the relevant literature.

*Part II* details the design, methodology, and implementation of the DeVAA Proof-of-Concept.

*Part III* presents the results of our experiments, discusses their implications, and concludes with future work.

***Chapter 2: Literature Review***

*(Primary Authors: Youssef Amjahdi, Abdelmounaim Sadir)*

**2.1 Autonomous AI Agents and Multi-Agent Systems**

*(Primary Author: Youssef Amjahdi)*

The concept of autonomous agents has evolved significantly with the advent of LLMs. This section reviews the foundational architectures that enable agents to reason and act.

Yao, S., et al. "ReAct: Synergizing Reasoning and Acting in Language Models." arXiv:2210.03629, Oct. 2022.

* ***Problem:*** Standard LLMs are poor at executing complex tasks that require planning and dynamic interaction with external tools.
* ***Method:*** The ReAct framework prompts the model to generate interleaved steps of reasoning ("thought") and action (e.g., an API call). This synergy allows the agent to create and execute dynamic plans.
* ***Relevance/Gap:*** This paper establishes the architectural pattern for the agents we intend to build. The gap is its inherent lack of verifiability; ReAct trusts the agent's reasoning. Our framework addresses this by demanding cryptographic proof of the "action" step's integrity.

LangChain AI. "Agents." LangChain Documentation, 2024.

* ***Problem:*** Building applications with LLMs requires significant boilerplate code to manage prompts, state, and tool interactions.
* ***Method:*** LangChain provides a standardized framework that abstracts this complexity, using an LLM as a reasoning engine to decide which "Tool" to use to accomplish a goal.
* ***Relevance/Gap:*** LangChain will serve as the SDK for our PoC agent, allowing us to focus on the novel ZKP verification layer rather than reinventing the agent's basic operational chassis.

**2.2 Decentralized Marketplaces & Reputation Systems**

(Primary Author: Abdelmounaim Sadir)

The creation of a functional digital economy requires mechanisms for exchange and trust. This section reviews prior work in decentralized marketplaces and the computational systems used to manage reputation.

Jøsang, A., Ismail, R., & Boyd, C. “A Survey of Trust and Reputation Systems for Online Service Provision.” Decision Support Systems, 43(2), 2007.

* ***Problem:*** Online platforms lack pre-existing trust, creating risk in transactions with unknown parties.
* ***Method:*** The paper surveys computational trust and reputation systems, defining key concepts, architectures, and vulnerabilities (e.g., Sybil attacks).
* ***Relevance/Gap:*** This survey is foundational for our reputation system. It highlights the challenges we must consider. The gap is its focus on systems with non-verifiable claims; our framework fills this by coupling reputation with cryptographic proof of actions.

Park, G., Jeon, B., & Lee, G. M. "QoS Implementation with Triple‑Metric Based AQM for Military Networks." Electronics, 12(1), 2023.

* ***Problem:*** Traditional network queue management often optimizes for single metrics, leading to poor performance under congestion.
* ***Method:*** The paper introduces a multi-metric approach that considers queue length, packet loss, and delay simultaneously.
* ***Relevance/Gap:*** This work validates the importance of metric-driven decision-making. Our work extends this concept by making the reported metrics themselves verifiable, ensuring the integrity of the data used for reputation and decision-making.

**2.3 Verifiable Computation and Zero-Knowledge Proofs**

*(Primary Author: Youssef Amjahdi)*

The core innovation of this thesis lies in applying verifiable computation to the agent economy. This section reviews the foundational technologies that make this possible.

Buterin, V. "Anatomy of a ZK-SNARK." vitalik.ca, Jan. 26, 2021.

* ***Problem:*** The inner workings of zk-SNARKs are notoriously complex, creating a high barrier to entry.
* ***Method:*** Buterin provides a pedagogical, high-level explanation of how zk-SNARKs are constructed, building an intuitive understanding of the core cryptographic concepts.
* ***Relevance/Gap:*** This post provides the essential conceptual understanding of the technology for our verification mechanism (RQ2). The gap is that it is purely educational; our thesis bridges this by implementing ZKPs for this novel application.

Circom Team. "Circom 2 Documentation." circom.io, 2023.

* ***Problem:*** Directly writing the mathematical constraints (R1CS) for a zk-SNARK is incredibly difficult and error-prone.
* ***Method:*** Circom is a domain-specific language for defining arithmetic circuits. The compiler handles the complex process of converting this logic into the format ZKP systems use.
* ***Relevance/Gap:*** Circom is the core technology for answering RQ2. It is the critical bridge between the AI agent's task and the cryptographic proof of its integrity, allowing us to build the ZKP without being cryptographers ourselves.

**2.4 Identifying the Research Gap**

The literature shows a clear convergence of powerful AI agents, decentralized economies, and verifiable computation. However, these fields have largely evolved in parallel. There is a distinct lack of research and practical implementation at the intersection of these three domains. No existing framework provides a mechanism for the cryptographically verifiable, trustless exchange of services from autonomous AI agents in a decentralized marketplace. This thesis aims to fill that specific and critical gap.

**Part II: The DeVAA Framework: Design & Implementation**

***Chapter 3: System Design & Methodology***

*(Primary Authors: Youssef Amjahdi, Abdelmounaim Sadir)*

**3.1 Introduction to Method**

This chapter provides the complete technical blueprint for the DeVAA framework. The methodology employed is one of a Proof-of-Concept (PoC), designed to be the simplest possible end-to-end implementation that directly addresses our three core research questions. The system is logically divided into two primary domains: the Off-Chain System and the On-Chain System.

**3.2 Off-Chain System Design: The Agent & Verifiable Computation**

*(Primary Author: Youssef Amjahdi)*

**3.2.1 AI Agent Task Logic**

To manage complexity, we define a simple, deterministic sentiment analysis task using a keyword-based scoring algorithm.

Input: A string of text.

Process: The agent calculates a score based on the presence of pre-defined positive and negative keywords.

Output: An integer: 1 for positive, -1 for negative, 0 for neutral.

This deterministic logic is essential for its representation within a ZKP circuit.

**3.2.2 The ZKP Circuit Design**

The core of our verification is a zk-SNARK circuit written in Circom.

Circuit Pseudo-Code (sentiment.circom):

Generated javascript

template SentimentVerifier() {

// Public Inputs

signal input text\_hash;

signal input sentiment\_result;

// Private Inputs

signal input text\_words[10];

// Logic

// 1. Verify text\_words match text\_hash.

// 2. Calculate score from text\_words.

// 3. Constrain that the calculated score's classification matches sentiment\_result.

}

Use code with caution.

JavaScript

**3.2.3 Key Mathematical Concepts: Quadratic Arithmetic Programs (QAP)**

A zk-SNARK represents a computation as a QAP, which checks if (A · s) ◦ (B · s) - (C · s) = 0, where s is the solution vector and A, B, C are matrices representing the program's rules. The ability to prove knowledge of a valid s without revealing it is the foundation of our verification mechanism.

**3.3 On-Chain System Design: The Marketplace & User Interface**

*(Primary Author: Abdelmounaim Sadir)*

**3.3.1 Smart Contract Architecture**

We will develop three core Solidity contracts:

AgentRegistry.sol: An ERC-721 contract to mint NFTs representing agent identities.

JobBoard.sol: Manages job lifecycles, acts as an escrow, and calls the Verifier.

Verifier.sol: An auto-generated contract from SnarkJS that verifies the ZKP.

**3.3.2 DApp User Flow & Wireframes**

A minimal React DApp will facilitate the user journey:

Connect wallet.

Post a job and payment to the JobBoard.

View the job status, which updates upon verified completion.

(Wireframes to be inserted here)

**3.3.3 Core On-Chain Logic**

The completeJob function in JobBoard.sol is the critical control point.

Pseudo-code for completeJob:

Generated solidity

function completeJob(uint256 jobId, bytes memory proof, uint256[] memory publicSignals) external {

// 1. Check job status and caller identity.

// 2. Call verifier.verifyProof(proof, publicSignals).

// 3. If true, mark job complete and transfer payment.

// 4. If false, revert the transaction.

}

Use code with caution.

Solidity

**3.4 The End-to-End Workflow**

The system operates via a user-agent-blockchain interaction loop. A user posts a job on-chain. An off-chain agent discovers the job, executes the task, generates a ZKP of its work, and submits the proof back on-chain. The smart contract verifies the proof and releases payment, completing the trustless exchange.

***Chapter 4: Proof-of-Concept Implementation***

(This chapter will be written during the coding phase)

4.1. Development Environment & Tech Stack (Abdelmounaim)

4.2. Implementing the ZKP Circuit (Youssef)

4.3. Implementing the Smart Contracts (Abdelmounaim)

4.4. Implementing the AI Agent (Youssef)

4.5. System Integration Challenges (Youssef, Abdelmounaim)

Part III: Evaluation & Conclusion

Chapter 5: Results and Evaluation

(This chapter will be written after the PoC is built)

5.1. Experimental Setup

5.2. Answering RQ1: Architectural Analysis

5.3. Answering RQ2: ZKP Generation & Verification Analysis

5.4. Answering RQ3: Performance & Economic Analysis (Gas Costs, Latency)

Chapter 6: Discussion

6.1. Interpretation of Results

6.2. Limitations of the Study

6.3. Security Considerations and Potential Attacks

Chapter 7: Conclusion

7.1. Summary of Contributions

7.2. Future Work

7.3. Concluding Remarks

Bibliography

(To be populated by Zotero)

Appendices

Appendix A: Source Code Listings

Appendix B: Installation & Setup Guide