EncryptAI: A marketplace for Privacy Preserving AI using Zero Knowledge and Fully Homomorphic Encryption

Master's Thesis
Blockchain Technologies

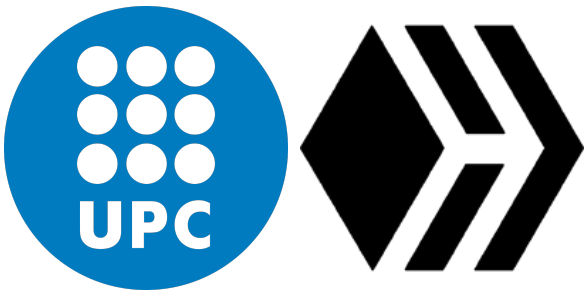
BY

Joan Flotats

Marcel Gerardino

Marc Llopart

Guillermo Oyarzun



UPC School
Barcelona - October 2024

Github

Abstract

With the increasing relevance of privacy in today's technological environment, Zero-Knowledge (ZK) proofs and Fully Homomorphic Encryption (FHE) are becoming the most popular ways to improve user privacy. EncryptAI uses these technologies to address issues with data accessibility, fairness, and monetization. Data has become a very valuable asset; nevertheless, monetizing and safely sharing it remains difficult, especially given the rising conflict between data privacy and the benefits of AI. To address this issue, EncryptAI proposes a Web3-based marketplace that allows data suppliers to distribute encrypted data using FHE, AI engineers to build models on this data, and users to make inferences on encrypted data using ZK-verified computations. This marketplace enables AI developers to train on encrypted user data, check computational correctness, and properly reward users for data usage, thereby encouraging a safe, traceable, and decentralized data economy.

Contents

1 Introduction

Since the introduction of large language models (LLMs), artificial intelligence has become a focus in a variety of societal sectors. However, advances in AI have resulted in an increased demand for massive volumes of data, frequently at the expense of users' privacy. Furthermore, centralizing AI services among a few significant suppliers increases the danger of bias and limited accessibility. To promote AI innovation while protecting privacy, EncryptAI provides a system that combines advanced encryption algorithms to enable secure, equitable data sharing without jeopardizing user anonymity.

EncryptAI is a privacy-first data marketplace designed to address these issues by integrating Zero-Knowledge (ZK) proofs and Fully Homomorphic Encryption (FHE). By leveraging FHE, AI engineers can develop and train models on encrypted data, ensuring the confidentiality of user information even during computations. Additionally, EncryptAI enables users to benefit from ZK-verified computations, which confirm the integrity of AI processes without requiring data decryption. Through this innovative combination of ZK proofs and FHE, EncryptAI offers a decentralized, secure, and verifiable solution for democratizing data in a privacy-preserving manner.

1.1 The EncryptAI Protocol

EncryptAI Protocol is a trailblazing protocol that combines cutting-edge cryptography techniques, decentralized infrastructure, and incentive systems to create a data marketplace with a privacy focus. Fundamentally, the EncryptAI Protocol integrates Fully Homomorphic Encryption (FHE) and Zero-Knowledge (ZK) proofs to facilitate safe data exchange and AI model training. By ensuring that data providers can exchange encrypted data, this two-layered strategy enables AI developers to carry out calculations without disclosing any private information. EncryptAI provides an extra degree of assurance by using ZK proofs, which guarantee that the calculations are carried out accurately and as planned while maintaining user anonymity.

EncryptAI's foundation on a decentralized blockchain network contributes to its privacy and accessibility goals. By deploying on an EVM-compatible chain, the protocol employs smart contracts to securely manage marketplace logic, data access rights, and transactions, ensuring trustless interactions between data providers and AI developers. To achieve efficient decentralization while maintaining operational integrity, EncryptAI employs a distributed node architecture in which each node conducts inference tasks and participates in data processing on encrypted datasets. These nodes are compensated by a staking mechanism that uses the protocol's native token, matching node operators' motives with the platform's purpose of ensuring data security and computation stability.

The EncryptAI protocol provides a ground-breaking answer to the AI ecosystem's present privacy and centralization concerns. EncryptAI enables data suppliers and AI engineers to work in a transparent, privacy-preserving environment, setting a new standard for secure and decentralized AI creation.

1.2 Goals of the Project

The primary goal of EncryptAI is to build a safe and scalable data marketplace using Fully Homomorphic Encryption (FHE) and Zero-Knowledge (ZK) proofs. This marketplace will enable data producers, AI engineers, and users to work together in a private, decentralized environment where data may be securely monetized and calculations verified. To accomplish this general purpose, EncryptAI is organized into the following specific objectives:

1. Market Research: Conduct a thorough analysis into existing data marketplaces and privacy-preserving technologies. This study seeks to identify gaps, best practices, and areas for development that will help shape the creation of a superior marketplace platform.

2. Understanding and Integrating FHE: Learn about the mechanics of Fully Homomorphic Encryption (FHE) before implementing secure data sharing and encrypted computations. Mastering Zama's [1] software stack will be a primary focus in order to maximize efficiency and security in data encryption.

3. Creating the ZK Proof System: Investigate and evaluate existing zero-knowledge proof methods in order to create a system that allows for ZK-verified calculations for AI models. This stage ensures data integrity and privacy by confirming computations without revealing the underlying data.

4. Implementing Business Logic: Create the smart contracts required to conduct marketplace operations and deploy them on a testnet environment. This includes creating and implementing a native coin to enable network incentives, transactions, and staking mechanisms built into the protocol.

By accomplishing these goals, EncryptAI hopes to create a decentralized, privacy-first data marketplace that serves the needs of AI-driven sectors while maintaining data security, transparency, and equitable access.

2 Basics of EncryptAI technical concepts

EncryptAI is based on complex technical ideas that support its goals of privacy, decentralization, and data security. Blockchain, Zero-Knowledge Proofs (ZK), and Fully Homomorphic Encryption (FHE) are key technologies that enable the EncryptAI marketplace. This chapter provides a simplified summary of each idea and explains how it relates to EncryptAI.

2.1 Blockchain

Blockchain is a distributed ledger technology (DLT) that maintains data in blocks [9][10], especially data from transactions, and is therefore by definition a DLT. As a result, the blockchain is decentralised since data is maintained across all network nodes, which are not controlled by any organization, records are immutable, and a set of rules called smart contracts that facilitate faster transaction processing are also provided.

Although EncryptAI is supposed to be chain-agnostic, this project was built on an Ethereum Virtual Machine (EVM)-compatible blockchain [2]. This decision allows EncryptAI to connect with the vast Ethereum ecosystem, gaining the benefits of a well-established smart contract platform. EncryptAI, as a chain-agnostic protocol, may be deployed on any blockchain that supports smart contracts, providing flexibility and adaptation across several blockchain networks.

2.2 Zero-Knowledge Proofs

A zero-knowledge proof (ZKP) is a protocol between two parties, a prover and a verifier, in which the prover makes a statement and attempts to persuade the verifier that it is true while revealing nothing other than the fact that the statement is true. In the context of EncryptAI, ZK Proofs verify that AI computations on encrypted data are accurate and trustworthy, exposing the data itself. This is especially beneficial for privacy-preserving AI since it allows secure verification of results without requiring access to raw data.

In this phase of the research, a zkVM for encrypted data was constructed to validate the outcomes of a regression problem using multivariate linear regression, as linear regressions are the foundation of AI challenges. However, this zkVM might be modified to include additional functionalities for classification problems and scale to larger models.

2.3 Fully Homomorphic Encryption

Fully Homomorphic Encryption (FHE) is an advanced cryptographic technology that enables calculations on encrypted data without the need for decryption. Unlike classical encryption, which requires data to be decrypted before operations can be conducted, FHE ensures data confidentiality throughout the computation process. This implies that users may securely exchange their data in encrypted form, and AI models can be trained or inferences drawn without ever revealing the raw data. In the context of EncryptAI, FHE allows data providers to securely communicate their data, while AI engineers can execute critical computations without jeopardizing privacy.

However, because FHE can be computationally costly, EncryptAI uses an optimized method known as Learning with Errors (LWE), a mathematical problem extensively employed in cryptography to create secure, partially homomorphic encryption systems. LWE-based encryption introduces a small amount of "noise" into encrypted data, making it more secure and resistant to illegal access. While it does not offer the entire range of calculations that FHE does, LWE strikes a practical compromise by allowing some sorts of operations—such as adds and limited multiplications—on encrypted data. By employing LWE, EncryptAI can efficiently handle homomorphic processes without the computing costs of complete FHE, making it more suitable for marketplace applications.

3 EncryptAI Initial Design

The EncryptAI marketplace's basic architecture is described in figure 1, along with how the platform plans to facilitate safe, private, and decentralized transactions utilizing AI models and datasets. Through the use of cutting-edge cryptographic algorithms, the architecture aims to ensure data privacy and model integrity while facilitating smooth interaction between several stakeholders, including data sources, model creators, and end users.

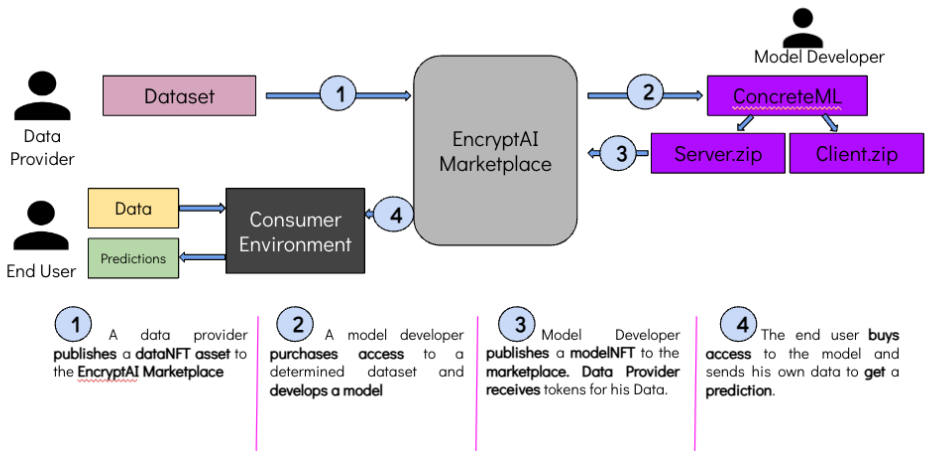


Figure 1: Schema of the initial EncryptAI design

The general idea of how EncryptAI marketplace would work is the following:

1. A data provider creates a **DataNFT** which would be an on-chain NFT representing the data source. This data must be already fully homomorphically encrypted and stored in a cloud solution such as IPFS. The access to this data source will be just granted once someone has purchased access to it.

2. An AI Engineer decides to purchase access to a data source and develops a model. This model is serialized and uploaded to IPFS.

3. Once the model is developed, this AI Engineer can create a **ModelNFT** which will be an on-chain representation of the model following the same appraoch of the **DataNFT**.

4. A user decides to run inference on a desired model, so the user will purchase a computation job with the desired model. This model will run on a protocol node and return the encrypted predictions along with a ZK proof.

3.1 Market Status

There are a number of new frameworks and solutions available in the current environment for safe AI model exchange and verification. The most pertinent ones are covered in this area, including platforms that provide validity proofs for AI calculations, homomorphic encryption libraries, and current blockchain marketplaces that seek to offer data sharing options.

- ZKML for AI Models: Giza [3] is a platform that focuses on Zero-Knowledge Machine Learning (ZKML), one of the first to apply zero-knowledge proofs to machine learning. Giza focuses on proving the correctness of machine learning calculations. One important lesson we learned from Giza's method is that before proofs can be calculated, machine learning models need to be serialized. Enabling ZK verification, protecting the integrity of the model, and avoiding tampering during execution all depend on this serialization procedure.

- Zama's FHE Library [5]: Computations on encrypted data are made possible by Full Homomorphic Encryption (FHE), while the technology is still developing. Leading homomorphic encryption company Zama offers a sophisticated library for working with encrypted AI models. Because completely homomorphic encryption is currently limited, EncryptAI uses Learning with Errors (LWE), a lattice-based encryption technique that enables safe calculations on encrypted data. LWE offers a workable solution for enabling safe AI processing on partially encrypted data, even though it doesn't provide the entire spectrum of FHE capabilities.

- Zero-Knowledge Proof Protocols: Although they provide frameworks for zero-knowledge proofs, projects such as SP1 [6][7] and RiscZero [8] are not made to function with encrypted inputs. EncryptAI seeks to solve the lack of support for privacy-preserving operations on encrypted data in these protocols, which concentrate on general-purpose ZK proofs and offer robust computational guarantees.

- Blockchain Data Marketplaces: Tokenized datasets are used in blockchain markets such as Ocean Protocol [4] to ease data exchange. The ASI Alliance also seeks to combine blockchain technology and artificial intelligence. However, due to their absence of reliable methods for managing encrypted data in a decentralized setting, both programs are constrained in terms of privacy-protecting features for the use and verification of AI models. EncryptAI sets itself apart in its marketplace by combining encrypted data management with privacy-preserving AI models.

Therefore, the efforts of this project have been more towards getting a Proof-of-Concept (PoC) of how the ZK part of the project should work instead of looking for the whole marketplace functionalities.

3.2 Validity Proofs: A simplified Version

EncryptAI has developed a simplified concept of a validity proof by leveraging the Concrete-ML library from Zama. This library is meant to be used for the development of models that can be trained and make predictions with FHE data.

The models from this library have been adapted to return in the inference step what has been called as a validity proof. The validity proof intends to prove that the proper model with the submitted input has been the one executed. However, this validity proof does not check for the correct computation of the model.

The process of how does it work is:

1. The serialized ML model is selected and ready to predict.
2. The model runs a prediction and computes a SHA256 with the sent id, the input and the model name extracted directly from the serialized file.
3. The returned proof is sent into the verifier to check if the predictions are from the selected model.

For EncryptAI, this method acts as an initial engagement with validity checking in machine learning. For the end user, it adds a layer of trustworthiness, making the process more transparent and trustworthy even though it doesn't provide all the guarantees of a full ZK proof. EncryptAI is also investigating a more thorough strategy by creating a zkVM (zero-knowledge virtual machine) for increased robustness. Stronger validity guarantees across all calculations would be excellent for this zkVM. These less complicated validity proofs, on the other hand, might be a more affordable and different choice in the EncryptAI market, giving consumers more options for the degree of verification they receive.

4 The Encrypt zkVM

The Zero-Knowledge Virtual Machine (zkVM) is the fundamental element of EncryptAI's technical architecture. It is intended to facilitate privacy-preserving computation on encrypted data by utilizing a combination of Zero-Knowledge proofs and lattice-based cryptography. Encrypted and unencrypted inputs can be processed safely in this zkVM's organized environment, guaranteeing data secrecy and computational integrity. This section discusses the zkVM's technical architecture, including its data handling procedures, encryption and decryption techniques, fundamental functions, and use of cryptographic hashing to ensure program integrity.

4.1 Technical Design

In order to enable encrypted calculations in a verifiable and secure way, the EncryptAI zkVM is built with particular inputs, encryption techniques, state machines, and hashing protocols. Note that only linear regressions and multivariate linear regressions can be proofed with this version.

Inputs

1. Public inputs: Non-encrypted inputs provided directly to the zkVM, accessible through the `READ` operation.
2. Secret Inputs: Encrypted inputs provided to the zkVM, accessible through the `READ2` operation. These inputs remain encrypted throughout the process to maintain data privacy.

4.2 Learning with Errors

EncryptAI's zkVM uses Learning with Errors (LWE), a lattice-based cryptographic technique, to encrypt data and perform secure encrypted operations. The LWE problem is built around the following equation:

$$A \cdot s = t$$

Where A is a public matrix, s is the secret key, and t is a public vector. In traditional terms, the secret key s can be derived through Gaussian elimination if both A and t are known. However, LWE introduces a random noise vector e to the equation:

$$A \cdot s + e = t$$

The noise makes it computationally infeasible to determine s accurately, even with access to both A and t, thereby securing the data.

4.3 Encryption and Decryption

To provide confidentiality, the zkVM's encryption and decryption procedures rely on this modified LWE principle.

4.3.1 Encryption

The zkVM creates a ciphertext with two parts during encryption:

- u: the cipher polynomial
- v: the polynomial vector

The encryption process follows:

$$u = t \cdot e_1 + e_2 + m$$
$$v = A \cdot e_1 + e_3$$

Where m represents the secret message, and e_1, e_2 and e_3 are random noise values. Together, u and v form the encrypted message, securing the data.

4.3.2 Decryption

To decrypt, the zkVM uses the secret key s with the following formula:

$$d = u - s \cdot u$$

Allowing therefore isolate the message m as follows:

$$d = m + e \cdot e_1 + e_3 - s \cdot e_2$$

With the noise values small enough, they can be rounded off, isolating the original message m for decryption.

4.4 Operations

The VM has the following operations implemented:

Operation	Definition	Opcode Value	Shift
PUSH	Push a value to the top of the stack	10_000	Right 1
READ	Read a value from public inputs and push it to the top of the stack	10_001	Right 1
READ2	Read a value from secret inputs and push it to the top of the stack	10_010	Right 5
ADD	Add two elements from the top of the stack	01_000	Left 1
ADD2	Add two ciphertexts from the top of the stack	01_011	Left 5
SADD	Add an element and a ciphertext from the top of the stack	01_010	Left 1
MUL	Multiply two elements from the top of the stack	01_001	Left 1
SMUL	Multiply an element and a ciphertext from the top of the stack	01_100	Left 1

Table 1: Operations Supported by EncryptAI zkVM

4.5 State Machines

The zkVM utilizes several state machines to handle its core functions.

1. **System State Machine**: Tracks the clock cycle for each operation.
2. **Stack State Machine**: Manages stack depth and tracks the stack's registers values, such as ADD and PUSH operations.
3. **Decoder State Machine**: Converts operation codes into binary representation, enabling translation and execution of commands.
4. **Chiplet State Machine**: Executes Rescue-Prime [11] hashing rounds, producing a trace with bits and operation values.

4.6 Program Hash

The Rescue-Prime hashing algorithm, a cryptographic sponge function, is used by the EncryptAI virtual machine (VM) to generate program hashes. The Rescue-Prime sponge function, which is useful for hashing and encryption, converts input data into a fixed-size output and was created especially for zero-knowledge proof applications. The procedure is broken down into two primary stages: the absorbing phase, in which the function receives input messages and uses specific mathematical operations to combine them with an internal state; and the squeezing phase, in which the internal state is further processed to yield a fixed-length output.

Key parameters for the Rescue-Prime function in EncryptAI zkVM include:

- State Length: 4
- Number of Rounds: 14
- Cycle Length: 16

Additionally, the solution streamlines integration with zero-knowledge proofs by optimizing arithmetization by absorbing elements between permutations instead of before applying them.

4.7 Arithmetization

The EncryptAI VM uses the Winterfell [12] proof system, which permits Algebraic Intermediate Representation (AIR) assertions, to easily produce zero-knowledge proofs. These AIR assertions use restrictions that must be met during execution to ensure that the virtual machine operates correctly. Important claims consist of:

- The **clock value** starts at 0.
- The **stack depth** also starts at 0.
- The initial **stack registries** and **hash values** are set to 0.
- The **final stack registries** and **hash values** match the program's expected output at the end of execution

4.8 Transitions

To guarantee that state changes between stages are valid, the VM uses particular transition restrictions. Each transition constraint has a polynomial degree based on the complexity of the operation, and all constraints must equal zero to preserve a valid state. Important transitions consist of:

- **Clock Increment**: Each operation increments the clock by 1.
- **Shift Operations**: Operations either shift left (Shl), shift right (Shr), or keep the stack as is.
- **Multiplications**: Ensures the stack's top value equals the product of two prevous values.
- **Rescue-Prime Hash**: The Program Hash uses periodic constraints. Periodic constraints ensure that certain values or conditions repeat over a predefined cycle. The Hash flag and ARK values are cyclic values that repeat over a cycle depending on the round step.

5 Smart Contracts

Smart contracts are crucial to the EncryptAI protocol's ability to securely and decentrally manage communication between data sources, model creators, and end users. The ownership and permissions of datasets, machine learning models, and transactional tokens on the EncryptAI marketplace are governed by these contracts. The three main contracts—the DataNFT contract, the ModelNFT contract, and the EncryptAIToken (an ERC-20 token)—are covered below. To further ensure a smooth and scalable process for developing new assets on the platform, there is the TokenFactory, which serves as a deployment factory for the DataNFT and ModelNFT contracts.

5.1 Asset Tokenitzation

Two ERC-721 contracts, DataNFT and ModelNFT, are used in the EncryptAI marketplace to tokenize datasets and machine learning models. By converting their assets into distinct, non-fungible tokens (NFTs), these contracts enable data providers and model developers to make their assets safe, verifiable, and transferable.

Key Functionalities:

- Data and Model Representation: Tokenizing datasets and models, respectively, is the goal of DataNFTs and ModelNFTs, which turn them into distinctive assets in the marketplace. Every NFT has metadata unique to the asset it represents, including descriptive names, related properties, and IPFS URIs for decentralized storage.
- Ownership and Control: By minting NFTs, which stand for ownership, data suppliers and model creators maintain control over their assets. In order to avoid duplication and guarantee that every item on the platform is unique, the contracts impose unique identifiers (such dataset hashes).
- Pricing and Licensing: Customized pricing models are supported by both varieties of NFTs. Every time an end user accesses the data or makes use of the model, data providers and model developers can profit from their assets by charging a usage fee or a one-time access payment.

EncryptAI enables transparent, equitable monetization in a decentralized marketplace while empowering data sources and model developers to safely share their work through asset tokenization. This method establishes a standardized and safe framework for trading and managing AI assets.

5.2 EncryptAI Token

The main currency in the EncryptAI marketplace is the EncryptAI Token, an ERC-20 utility token. This token enables smooth payments for asset access and usage by facilitating transactions between data providers, model developers, and end users.

Key Functionalities:

- Transaction Currency: EncryptAI Token is used for purchasing access to DataNFTs and ModelNFTs, acting as the medium of exchange within the platform.

- Unified Marketplace Currency: As a single token supporting all transactions, EncryptAI Token simplifies the marketplace's economics, providing a consistent and straightforward experience for all users.

By utilizing a native ERC-20 token, EncryptAI ensures a unified and efficient transaction layer, aligning incentives across the ecosystem while supporting seamless interaction between different assets.

5.3 Token Factory

To facilitate the development and implementation of DataNFT and ModelNFT contracts, a smart contract called the Token Factory was created. The Token Factory streamlines the tokenization assets creation by serving as a central deployment manager.
Key Functionalities:

- Centralized Minting: The factory pattern allows for a standardized process for creating DataNFTs and ModelNFTs, reducing complexity for data providers and model developers who wish to tokenize their assets.

- Management and Access Control: The Token Factory can enforce permissions and access control policies, ensuring that only authorized users can deploy new contracts or interact with certain marketplace functions.

- Scalability: The Token Factory makes it easy to create new instances of NFTs as needed, supporting the growth and evolution of the EncryptAI marketplace.

Through the Token Factory, EncryptAI achieves a flexible, efficient deployment system, enabling rapid expansion of tokenized assets while maintaining security and standardization.

6 Conclusions and Future Work

In a nutshell, The basis for a decentralized marketplace centered on privacy-preserving AI products has been strengthened by the development of EncryptAI. However, because of the difficulty of building a zero-knowledge proof (ZKP) system that can handle the particular requirements of machine learning (ML) models, this first version focuses exclusively on regression problems.

Integrating Fully Homomorphic Encryption (FHE) with ZKPs has been one of the project's main problems. It has proven difficult to combine these two cutting-edge cryptography techniques since they have differing computational needs and underlying mathematical features. This intricacy emphasizes how creative yet difficult it is to put safe, private AI technologies into practice.

Since the zkVM is now modified to operate with `uint8` data types, one of the system's limitations is how numerical accuracy is handled. Since many machine learning tasks require fine-grained numerical operations, this constraint makes it challenging to directly understand signed numbers. Some applications are limited by this precision requirement, especially those that call for greater numerical accuracy or the usage of signed numbers in intricate computations.

The validity proofs, not the ZK proofs, created for this version are flexible and applicable to any model that is compatible with the Concrete ML library, despite these drawbacks. Even while this proof is not yet as thorough as a full zero-knowledge proof, it gives end users a certain amount of security by enabling them to confirm that the model that was executed was, in fact, the intended model.

EncryptAI needs to advance in a few areas before it can become a fully functional marketplace for private AI. The first step is to improve the zkVM's capabilities. By creating a more reliable zkVM that can handle signed integers and has higher numerical precision, the system will be able to support a wider variety of machine learning applications. EncryptAI would be able to support more AI models and serve a broader range of use cases if the VM were modified to perform classification tasks like Logistic Regression.

The EncryptAI Token requires a well-defined tokenomics structure. The ecology of the platform will be strengthened by establishing explicit incentives for end users, model creators, and data producers. Staking and reward systems will be required to promote participation and guarantee fairness. Lastly, the integration of inference nodes that can manage model execution in a distributed fashion will be necessary to achieve complete decentralization. Currently, this is a pain point due to the requirements for larger models. Staking and prizes for these nodes will encourage wider involvement and enhance the platform's scalability and security. When combined, these improvements will support EncryptAI's development into a strong, decentralized marketplace for AI applications that preserve privacy.

All in all, EncryptAI has achieved a lot of its initial objectives, creating the framework for a decentralized market that places a high priority on data security and privacy in AI. Significant progress has been achieved in tokenizing assets, creating cryptographic assurances, and creating the enabling contracts for an AI marketplace that protects privacy, even though the platform is not yet production-ready. The EncryptAI project lays the groundwork for future developments that will move the field closer to a fully functional and scalable solution while offering a promising framework for safe, decentralized AI.

A Appendix

The specific parts of the project and the code pieces can be found under the official EncryptAI repository. The particular parts of the project can be found on their own repository.

- concrete-ml: Public repo that is a fork from Zama's original repo. Contains the validity proof implementation along with its docker image to run inference.
- Encrypt-zkVM: Private repo containing the implementation of the zkVM.
- dApp: Public repo with the marketplace code.
- contracts: Public repo containing all smart contracts deployed.

References

[1] Zama. (2024). ZAMA. https://www.zama.ai/

[2] Ethereum. (2024). Ethereum Virtual Machine (EVM). https://ethereum.org/ca/developers/docs/evm/

[3] Giza Tech. (2024). Giza. https://www.gizatech.xyz/

[4] Ocean Protocol. (2017). Ocean Protocol. https://oceanprotocol.com/

[5] Github. (2024). TFHE-rs. https://github.com/zama-ai/tfhe-rs

[6] Succint. (2024). SP1. https://docs.succinct.xyz/

[7] Github. (2024). SP1. https://github.com/succinctlabs/sp1

[8] Github. (2024). risc0. https://github.com/risc0/risc0

[9] Wikipedia Foundation. (2023). Blockchain. https://en.wikipedia.org/wiki/Blockchain

[10] IBM. (2023). Blockchain success starts here. https://www.ibm.com/topics/blockchain

[11] Rescue-Prime. (2020). Rescue-Prime: a Standard Specification (SoK). https://eprint.iacr.org/2020/1143.pdf

[12] Github. (2024). Winterfell. https://github.com/facebook/winterfell