Zero-Knowledge Proofs for Machine Learning Integrity

Alexander Baron

Department of Computer Science
California State University, Chico
Chico, USA
ajbaron@csuchico.edu

Abstract

Large Language Models (LLMs) are trained on datasets that contain sensitive information, including personal data, proprietary documents, and confidential communications. Organizations often need to demonstrate their models' outputs while maintaining the confidentiality of their training data. Existing verification approaches require either full dataset disclosure or reliance on trusted third-party systems, creating privacy risks. This work proposes a framework using zero-knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARK) to cryptographically verify LLM output without revealing the training data. In order to validate this approach we implement a proof-of-concept on a small-scale language model. We aim to enable trustworthy AI deployment in privacy critical applications by providing cryptographic guarantees of model integrity without compromising data confidentiality.

Introduction

In 2017, researchers from Google published "Attention is all you need", introducing the Transformer architecture to the world. Since then, Transformers have been a major driver of the recent AI boom. As Transformers are increasingly utilized, more sensitive data will be used to train models. Consequently, protecting both input data and model weights is becoming increasingly essential.

Traditional approaches either expose model weights, or require trusting the server. Using zk-SNARK it is possible to prove that a given output was produced by a specific model without revealing the input data or the model weights. However applying zk-SNARK to verify model computations presents significant challenges.

Modern large language models often have billions of parameters and translating their forward pass into arithmetic circuits results in enormous constraint systems, rendering zk-SNARK proof generation computationally infeasible. To prevent this we adapt a compact Transformer based architecture that omits activation functions, and employs a very minimal attention head. This will reduce the complexity of the resulting arithmetic circuits and proof times.

System Design & Architecture

Our zk-SNARK workflow utilizes a small Transformer based LLM with a vocabulary size of 100, hidden dimensions of 32, and a context length of 8, totaling ~12,500 parameters. The model features a minimal attention head and omits activation functions, while providing a forward pass to compute the expected output.

The circuit input generator pads token ID's and position embeddings to match the model's sequence length. It then converts all the model weights to 18-bit fixed-point integers and generates *input.json* for Circom witness generation. A commitment to the model weights is computed using a Poseidon style hash, ensuring integrity without revealing the weights.

The Circom circuit implements a simplified Transformer forward pass including embedding lookup, positional embedding addition, linear attention, feed forward network, and output projection. The circuit's output is the predicted token, which will serve as the basis for the zk-SNARK proof.

The workflow is orchestrated by a program integrating all components. Starting by compiling the Circom circuit, and generating R1CS and WASM files. Next, a witness is generated using **Node.js** populating it with circuit inputs. The trusted setup is then prepared, producing the proving and verification keys. A zk-SNARK proof is generated as a JSON file, along with the corresponding public signals, and the correctness of the proof is subsequently verified.

Libraries and Tools

For the Python scripts the following libraries were used, **NumPy** for numerical computations, matrix operations, and fixed-point conversions. **JSON** for reading and writing circuit inputs, witness files, proofs, and public signals. **SHA256** for computing a Poseidon style hash of model weights for commitment. **Subprocess, os,** and **shutil** for programmatically executing Circom and SnarkJS commands and managing files.

For our zk-SNARK work flow we utilized **Circom 2.0** for constructing arithmetic circuits representing our Transformer architecture. **SnarkJS** for managing powers of tau, generating proving and verification keys, and verifying proofs. **Node.JS**, which is required for executing Circom generated witness scripts.

Results

All the experiments were conducted on a MacBook Pro with Apple M2 chip. Our Python Transformer model successfully generated a predicted token for the input [12,3,89,1], with the top 3 predictions being [1,0,61]. The circuit compiled successfully, and our witness was generated without any errors. The following output was produced after compiling the Circom circuit.

```
template instances: 12
non-linear constraints: 95648
linear constraints: 67836
public inputs: 0
private inputs: 14856
public outputs: 1
wires: 178341
labels: 429498
Written successfully: ./zk-SNARK.r1cs
Written successfully: ./zk-SNARK.sym
Written successfully: .
/zk-SNARK_js/zk-SNARK.wasm
Everything went okay
```

We have ~14,000 private inputs, and 1 public output suggesting that the circuit successfully incorporated the model weights and input data as private inputs, while exposing the predicted output as the single public output. The proof size was 742 bytes demonstrating the succinctness of the zk-SNARK approach.

The Powers of Tau, and Groth16 required significant computation, taking over an hour to complete, however these can be reused for generating additional proofs, reducing the cost of future computation.

We successfully generated a zk-SNARK proof for our Transformer model. The **Proof JSON** containing the Groth16 proof, and **Public Signals** containing the commitment to the model weights. The proof was verified successfully confirming the correctness of the zk-SNARK computation

Security & Correctness

Our system's security relies on the properties of the Groth16 zk-SNARK protocol and the trusted setup. The Powers of Tau ceremony ensures a secure multi-party computation, preventing a single participant from generating malicious proving or verification keys. Once the proving and verification keys are created, they can be safely reused without compromising security.

Correctness is ensured by accurately implementing the Transformer forward pass. The proof generation guarantees that the predicted token corresponds to the private inputs and model weights. Verification of the proof confirms the integrity and correctness of the computation, with any tampering resulting in verification failure.

Soundness is guaranteed by the Groth16 protocol's cryptographic assumptions. Specifically, the knowledge-of-exponent assumption in pairing-based cryptography ensures that a valid proof can only be generated by someone who possesses a valid witness satisfying all circuit constraints.

Conclusions

We demonstrated that zk-SNARK can be applied to Transformer-based models. However, the computational cost of generating proofs increases significantly with model size. Our simplified Transformer with ~12,500 parameters required over an hour to complete the trusted setup and generate the initial proof. Scaling to models with billions of parameters would likely take days or even weeks.

Future work could focus on leveraging more powerful hardware to reduce computation time. Additionally, exploring alternative architectures such as State Space Models or Mixture-of-Experts may offer efficiency gains and lower proof generation costs, making zk-SNARK integration more practical for larger models.