zkHub: Accelerating Privacy-Preserving Zero-Knowledge Proof Generation for AI Defense

Introduction

This proposal presents zkHub, a groundbreaking platform for accelerated generation, efficient management, and innovative application of zero-knowledge proofs (zk proofs). Built upon the most advanced cryptographic techniques and system optimizations, zkHub's architecture promises to deliver superior proof generation speed while ensuring exceptional levels of privacy and security. Importantly, this platform introduces several pivotal components, such as Developer SDK, Client SDK, Node Runner Package, Backend SDK, Proof Repository, and robust testing tooling. Furthermore, it aims to integrate zk proofs into a variety of cyber defense applications, providing a robust mechanism for AI-driven security measures, such as identifying source code vulnerabilities, optimizing confidential computing on GPUs, and automating incident triage.

Set of Questions or Problems the Project Hopes to Address

- How can we substantially accelerate zk proof generation while maintaining rigorous privacy and security standards?
- How can we leverage zk proofs in applications like automating incident triage, identifying security issues in source code, patching vulnerabilities, and optimizing confidential compute on GPUs?
- How can we integrate zk proofs into AI-driven cyber defense mechanisms while ensuring that the computational burden does not overwhelm user machines?

Problem Statement

Ensuring Integrity of AI Systems: Currently, users face several challenges related to AI and machine learning models. They lack a secure method of verifying a model's authenticity, resulting in the risk of being served inferior or inaccurate models behind APIs. Moreover, there's a need to ensure that the same machine learning algorithm runs consistently on different users' data, devoid of arbitrary bias - an issue particularly pertinent in sectors like finance or healthcare. Another significant problem lies in integrating attestations from externally verified parties into a model, assuring the data originates from legitimate sources. The ability to perform machine learning inference or training in a decentralized way, enabling public model submission and data privacy is also lacking. Lastly, a method to verify someone's unique personhood without compromising their privacy is crucial in today's digital world, but the current systems fall short. These problems necessitate an efficient, privacy-preserving solution to enhance the integrity and credibility of AI systems.

Speed and Efficiency: Traditional methods for generating zk proofs are slow and computationally intensive. This limits their usability in real-world applications, especially on low-power devices such as smartphones and laptops. zkHub aims to significantly improve the speed and efficiency of zk proof generation.

Security and Privacy: As zk proofs become more prevalent in various applications, ensuring their security and privacy is critical. zkHub seeks to leverage zk proofs in a way that ensures privacy and security without sacrificing performance.

Developer Experience: Current methods for testing and deploying zk proofs are cumbersome and slow. The zkHub platform aims to simplify the developer experience by shifting the computational burden from user machines to the cloud.

The Intersection of AI and Zero-Knowledge Proofs

Zero-Knowledge Proofs (zk) are cryptographic methods primarily used in blockchains for scaling compute-constrained networks and protecting user privacy. The potential of zk extends beyond blockchain, most significantly intersecting with AI. Here's how:

- zk offers a trustless and verifiable computational system, which can assure integrity in ML models. Ongoing research to reduce memory footprints and enhance performance now allows certain AI algorithms to be verified on-chain.
- zk has a crucial role in decentralizing AI and AI. In an era where machine learning models are controlled by a limited group of people, zk provides a means to build systems in a way that allows transparency, user ownership, and privacy.
- zk can address concerns regarding the integrity of ML models. It can ensure that the ML model claimed to have been run is the one that actually ran, verifying correct computation and protecting data privacy.
- The scope of zk applications in AI is increasing. These include ensuring model authenticity, verifying model integrity across different users' data, performing decentralized AI training, integrating attestations, and providing proof of personhood without violating privacy.
- zk technology is still evolving and the practicality of computations that can be verified on-chain is expanding, allowing for broader applications in AI.

Novel Implementations of zkHub in AI Cyber Defense Applications

1. **Confidential Compute Optimization on GPUs**:

    - **Challenge**: Currently, the optimization of confidential computations on GPUs is a tricky process. It can potentially breach privacy because, without proper safeguards, sensitive data can be exposed during computations.
    - **zkHub Solution**:
        - **Verifiable Computation**: zkHub leverages zk-SNARKs, a form of zero-knowledge proofs, to enable verifiable computation. These proofs allow for the authentication and verification of computations without exposing the inputs or the process, maintaining the privacy of the data.
        - **Confidentiality of Computation**: Besides verifying the computations, zk-SNARKs also secure the data being processed. This is achieved by

creating a 'proof' that can be independently verified without exposing any specific details about the computation or the underlying data.
    - **Performance Optimization**: zkHub's protocols with zk-SNARKs optimize GPU resource utilization by allowing computations to be verified without re-execution, reducing redundant computational processes. This means that the GPU resources can be directed toward new tasks instead of repeatedly verifying past computations.
    - **Data Integrity**: zk-SNARKs provide a solid mechanism to maintain the integrity of data during AI computations. The verification of computation correctness ensures that any data manipulation or errors do not influence the final results.

2. **Source Code Vulnerability Identification**:

    - **Challenge**: In software development, identifying vulnerabilities within source code is a critical but challenging task. Developers often face the dilemma of potentially exposing sensitive information about their codebase during security analyses. This can lead to inadvertent disclosure of confidential data or intellectual property, thereby putting the entire project at risk.
    - **zkHub Solution**: zkHub leverages the unique properties of zk-SNARKs to engineer protocols that allow for secure, privacy-preserving verification of security analyses on source code. With the use of zk proofs, these protocols are capable of confirming the accuracy of the security analysis, giving developers invaluable insights into potential security risks.

3. **Patching Vulnerabilities**:

    - **Challenge**: Patching software vulnerabilities is often a cumbersome task that is fraught with risks. The process of applying patches can potentially lead to privacy breaches and there is often insufficient assurance that the patch has been correctly installed.
    - **zkHub Solution**: zkHub introduces a revolutionary approach to this process by utilizing zk-SNARKs. The platform provides zk proofs for verifying the accurate installation of patches, resulting in an automated, secure, and confidential patching process.

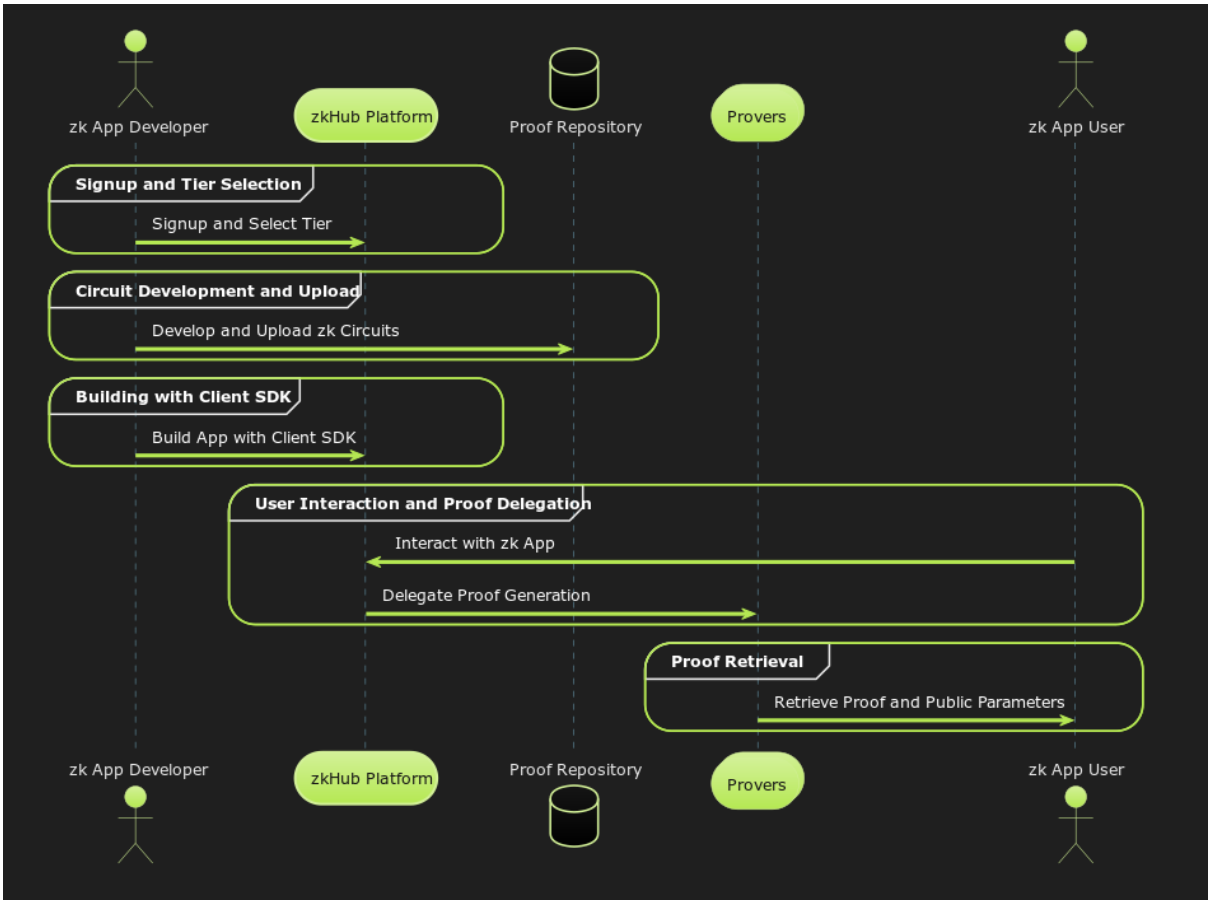
4. **Incident Triage Automation**:

    - **Challenge**: In incident response, administrators and analysts often have to work with sensitive data, which can risk potential misuse or accidental disclosure. The task of incident triage, where incidents are analyzed to determine their impact and prioritize response, can be particularly challenging in this regard.
    - **zkHub Solution**: zkHub addresses this issue using zk-SNARKs. The platform facilitates a privacy-focused incident response system by authenticating automated triage processes. This approach ensures that valid threats are identified and managed appropriately while maintaining the confidentiality of the incident data.

Methodologies and Approaches

Our product comprises four essential components that combine to facilitate the generation of proofs at an unprecedented speed. These components include the Client SDK, Developer SDK, Node Runner Package, and Backend SDK.

User Journey Workflow



1. **Signup and Tier Selection**: zk App Developers start their journey by signing up onto the zkHub platform. They select their preferred tier and price structure for proof generation based on their circuit size, allowed latency, and Quality of Service (QoS). zkHub also offers a free usage tier as a trial for developers.

2. **Circuit Development and Upload**: Developers then design their zk circuits and upload the compiled circuits to the zkHub Proof Repository. The circuits are uploaded in an intermediate representation format like R1CS (Rank 1 Constraint System), which is a common format for representing zk circuits.

3. **Building with Client SDK**: Developers build their applications using the zkHub Client SDK. The SDK, written in WebAssembly (WASM), supports Multi-party Computation (MPC) based proof delegation. It efficiently creates an MPC-based secret shared witness, which is then sent to the zkHub proof generation platform.

4. **User Interaction and Proof Delegation**: Users of the zk app interact with the application and delegate the proof generation to the zkHub platform. This delegation allows for faster proof generation and supports massive circuits. There are two configurations:
    - one where the proof delegator (end user's device) needs to be online for some interactions with the zkHub platform.
    - another where everything happens completely on the zkHub platform with some initial preprocessing on the user's device.

5. **Proof Retrieval**: The user retrieves the generated proof along with the public parameters, without leaking any secret witness data. This proof and parameters can be stored on the zkHub platform for further reuse if needed. This completes the user's journey, providing them with a secure and efficient solution for generating and managing zk proofs.

SDKs/Libraries at various levels

Developer SDK

The Developer SDK assists developers in testing and deploying zk-circuits.
Current methods of testing are slow, and limited by the computational capacities of the developers. zkHub aims to enhance the developer experience by shifting the burden of testing from user machines to the cloud, resulting in accelerated proof testing and allowing developers to keep their machines idle during this process.

Additionally, the Developer SDK will incorporate user-friendly tools that are crucial for deploying these circuits to the zkHub network.
It will offer multiple tiers catering to both individuals and businesses, each with distinct variations in limits, proving times, and analytics.

On the deployment of a proving circuit, the Developer SDK will store the circuit specifications in the **circuits repository.** Developers also get to choose privacy and speed requirements for their applications.

At launch, we have plans to provide support for circuits written in Arkworks and Circom.

Circuits Repository

The Proof Circuit Repository is a vital component of the zkHub platform, storing zk-circuit specifications in a centralized location. When developers deploy a zk-circuit via the Developer SDK, the specifications, usually in R1CS format, are securely stored in the repository. This facilitates efficient retrieval during proof generation, adhering to developers' privacy and speed requirements.

Additionally, the repository promotes optimization by enabling circuit reuse, thereby enhancing the overall efficiency and scalability of the zkHub network.

Client Side SDK

zkHub's Client SDK empowers developers to easily incorporate proving circuits into their applications, such as dapps, zkML apps, games, and more. Here's a streamlined overview of how it functions:

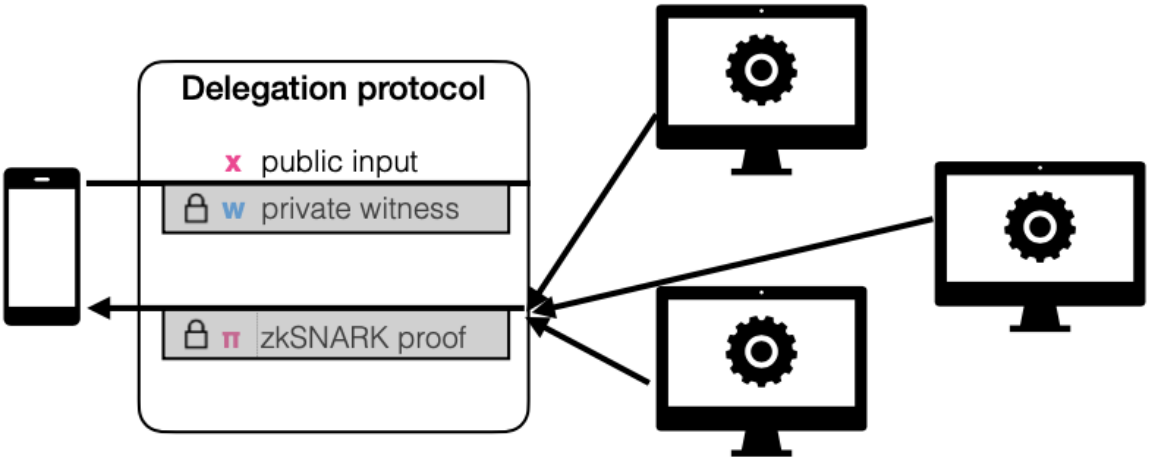
Proof Generation: The SDK communicates with the proof repository and zkHub network to access circuit details and information about available machines for proof creation.

Multiparty Computation (MPC) Initiation: The SDK employs the SPDZ protocol for its multiparty computation, optimized for both security and performance. The computation is represented via basic arithmetic operations, like addition and multiplication, which form the foundation of secure computations.

Efficiency in Randomness Generation: Unlike standard use cases of SPDZ protocol, zkHub's application reduces overheads significantly. This is achieved by generating the required randomness for multiplication computations on the client side, eliminating the need for expensive public-key cryptography.

Use of Beaver Multiples: The generated randomness, also known as Beaver Multiples, is distributed among the participating nodes, ensuring computation privacy. Each node receives a share of the Beaver triple (a, b, c) but doesn't have access to complete values.

Proof Verification: Post-generation, the Client SDK confirms the proof's accuracy before acceptance.



The zkHub Client SDK is crafted in WebAssembly (WASM) to ensure high performance and portability, allowing for operation in a variety of environments, including web browsers and standalone virtual machines.

Node Runner Package

The Node Runner package is a runtime software installed on the hardware that is to be plugged into the zkHub network. The node runner package is responsible for the following -

- Running benchmarking on hardware to determine capabilities.
- Accepting requests from the client and establishing a connection with the other nodes to begin generating distributed proofs.
  There are multiple inexpensive checks performed during the generation of the proof to ensure that each node is generating the proof honestly.
- The proof generators running on the nodes are optimized for MPC

Healthy nodes receive a better rating. (Higher-rated nodes generate proofs for organizations and enterprises)

Backend SDK

The Backend SDK manages the overall operations of the zkHub platform:

- Handles transaction processing, consensus mechanism, and interaction with the proof generation network.
- Provides APIs and tools to the Client SDK for efficient management and interaction of applications.
- Features for storing compiled circuits for reuse and providing analytics and alerts for system monitoring.

Project Roadmap

Q3 2023

- Develop the MPC x ZK SDK, Developer SDK, and Marlin proof system support.
- Begin the production of educational content.
- Add Circom Support.
- **Develop tooling for identifying security issues with zk proofs.**

Q4 2023

- **Develop tools for program analysis and proving software vulnerabilities.**
- Develop the Client SDK and establish marketplace mechanics.
- Develop ZK course and Groth16 proof system support.
- Release the Node Runner Package and Halo2 proof system support.
- Develop a CI/CD tool.
- Begin outreach campaigns.
- Publish the technical whitepaper.

Q1 2024

- **Develop zkML tooling for model integrity & API inferences.**
- Add support for Plonk and recursive proof systems.
- Create a proof repository.
- Organize a zkHub Hackathon.
- Begin optimizing processes and reducing costs.

Q2 2024

- **Develop tooling for proving the integrity of GPU computing & incident triage automation tools.**
- Expand to Aleo Network after rigorous security audits.
- Start the next round of fundraising.
- Expand the team.

- Develop a benchmarking toolkit.
- Begin work on hardware acceleration on FPGAs/ASICs.

Expected Results

Through the implementation of this proposal, we aim to:

1. Accelerate the generation of zk proofs significantly, with benchmarks demonstrating a speed-up of over 30X and an ability to handle circuits that are 250X larger compared to traditional methods.
2. Integrate zk proofs into a wide range of applications, from automating incident triage to identifying security vulnerabilities in source code, patching vulnerabilities, and optimizing confidential compute on GPUs, thereby enhancing the privacy and security of these applications.
3. Cultivate a vibrant community of zk-proof developers and users by providing a user-friendly platform that simplifies the development and deployment of zk-proofs.

Conclusion

zkHub represents a major step forward in the utilization of zk proofs. By accelerating zk proof generation and integrating them into various AI-driven cyber defense applications, it stands to revolutionize both the fields of cryptography and cyber defense.

Bibliography

1. Alex Ozdemir et al., "Fast, Scalable, and Communication-Efficient Zero-Knowledge Proofs Working with a Trusted Dealer," Cryptology ePrint Archive, Report 2021/499, 2021.
2. Alessandro Chiesa et al., “EOS: Efficient Private Delegation of zkSNARK Provers,” USENIX Security Symposium, 2022.
3. Srinath Setty et al., "Taking Proof-Based Verified Computation a Few Steps Closer to Practicality," USENIX Security Symposium, 2012.
4. Pratyush Mishra et al., "ZEXE: Enabling Decentralized Private Computation," IEEE Symposium on Security and Privacy (SP), 2020.
5. Howard Wu et al., "DIZK: A Distributed Zero Knowledge Proof System," USENIX Security Symposium, 2018.
6. Sean Bowe et al., "Halo: Recursive Proof Composition without a Trusted Setup," Cryptology ePrint Archive, Report 2019/1021, 2019.
7. Alessandro Chiesa et al., "Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS," IEEE Symposium on Security and Privacy (SP), 2020.
8. Maciej Skorski, "Zero-Knowledge Proofs: An Illustrated Primer," IEEE Security & Privacy, vol. 17, no. 6, pp. 37-45, 2019.
9. Eli Ben-Sasson et al., "Scalable, transparent, and post-quantum secure computational integrity," IACR Cryptology ePrint Archive, 2018.