OZKi
Open Zero Knowledge Integration

Capstone Project:
MICS Summer 2022

Aug 3, 2022 | 5:08 P.M.
Berkeley School of Information
Meet the TEAM

Lauren Ayala  
Project Manager

Suvojit Basak  
Database Admin

Antony Halim  
Software Architect and Developer

Mariah Martinez  
Software Developer
The Personally Identifiable Information Problem

Ongoing issues surrounding the controlling and securing of Personally Identifiable Information (PII):

- Billions of dollars lost to PII data breaches
- Commercialization of PII presents both ethics and security problems
- Current model of PII flow is broken
  - Users are asked to submit various PII
- No effective tools for Web2 app developers to mitigate PII leaks
Vision

- Build a very effective privacy protecting software component for the traditional web services.
- Password-less environment to identity less world.

Opportunity

- Verifiable Computation (aka ZKP)
- Use of ZKP has remained within the blockchain or Web3 applications
- No known ZKP-based developer's tool for building general purpose privacy protecting module for the Web2 applications
Zero Knowledge Proof

Secure Data and Proofs

Input: 3,4, 6,2, 12,1
Output: 12

Prover
Verifier

- Zero-Knowledge
- Completeness
- Soundness
What is OZKi?

OZKi is a ZKP based framework for developing privacy protecting components of a web application using verifiable computation as its underlying technology.
OZKi Components

OZKi has two main components:

- **OZKi Toolkit**
  - Toolkit CLI
    A command line tool to build a proof function. The *proof function* is the function where you define the app-specific constraints or requirements on the user’s PII.
  - Toolkit Library
    The library implements the typescript abstraction around the proof function by providing the *ProofGenerator* and *ProofVerifier* classes. This minimizes ZKP exposures to the web developers.

“OZKi is specifically designed to assist Web2 developers implement privacy-protecting software components with minimal efforts and without the need to use blockchain.” - OZKi Team
OZKi Components

- **OZKi Oracle Service**
  - Is a Web3 concept that we borrowed and used for the OZKi framework.
  - The oracle is used to bring external data into the ZKP computation.
  - OZKi designed the oracle to pull PII securely from the original sources.

The toolkit and the oracle work together to provide a secure end-to-end proving system.

“OZKi is specifically designed to assist Web2 developers implement privacy-protecting software components with minimal efforts and without the need to use blockchain.” - OZKi Team
OZKi enables the concept of **proof-based authorization**, which can be used to replace the traditional login process. In this model, the server receives only the proof, not the PII. Here are some of the main implementations:

- **Proof of Payment**
  The server can verify if the user has made the required payment to access the service without knowing the user’s PII.

- **Proof of Login**
  The server can verify if the user’s login properties match some conditions without knowing the email address itself.

- **Proof of Hash**
  In game scenario: the (leaderboard) server can verify if the user’s answer is correct without knowing the answer itself.

“OZKi is specifically designed to assist Web2 developers implement privacy-protecting software components with minimal efforts and without the need to use blockchain.” - OZKi Team
OZKi-Bot is a sample application that demonstrates the use of the OZKi-Toolkit to build proof-based authorization flows within web applications that make use of user-generated content for end-users referred to by developers.
Top Three Threats

Based on the threat modeling on OZKi, we have identified the following top three threats:

1. Bad or malicious input to the proof function
2. Unsafe proof function codes
3. Replay attack on the proof & its output
OZKi Framework Overview

Proof TTL

OZKi proving system has the concept of proof TTL (time to live). This is an important security feature which is used to determine the age of a proof.

Timestamp

To implement the proof TTL concept, every OZKi proof must have a timestamp at the time of proof generation. The timestamp is the number of milliseconds since January 1, 1970. Every OZKi proof function must take a timestamp input parameter, which is passed in by the caller. The timestamp value is also passed to the timestamp output parameter so that the verifier can check the proof timestamp.

The timestamp mechanism applies to both signed and unsigned proof. In the case of signed proof, the oracle service sets the timestamp. In the case of unsigned proof, the user sets the timestamp.

Age and Expired Proof

The age of a proof is therefore the delta of the current time and the proof timestamp. A proof with age which has exceeded the TTL is considered an expired proof. An expired proof will fail the proof verification process. By default, the OZKi proof verifier sets the TTL to 5 minutes.

Once a proof is generated, the proof can be used exactly once before it expires. OZKi will detect and refuse proof which has been used more than once before it expires. OZKi proof verifier keeps a small database which temporarily stores proofs which have been successfully verified.
Guideline

This is the guideline to implement proof-based authorization logic.

From a high-level view, four entities are involved in the proof-based authorization flow: the user (client), the server, the oracle, and the PII source. Overall, the flow should follow the sequence call illustrated by the diagram below.
ProofGenerator class

This class is defined in ozkilib and is used by the user (proven) to create a zk-snark proof which has been signed by the oracle service.

The ProofGenerator is an abstract class which is defined below. This is the type used to generate a signed proof.

```java
export default abstract class ProofGenerator<Type extends zk.
constructor(
  zkComponentPath: string,
  zkComponentName: string,
) {
  super(zkComponentPath, zkComponentName);
}

// the subclass needs to implement this method
// to format caller-specific input parameters
protected formatInput(customInput: Tipo
protected formatInputs(zkInputSignature: Unblinded
... deleted for clarity...

// generateProof - async function
function generateProof(customInput: Type

There are two simple steps to use this class:

1. Subclass the ProofGenerator class with your own specific `<Type>`
   that defines the custom input which you use for the circom prove

ProofVerifier class

This class is defined

This is the class used to verify the proof generated by ProofGenerator object.

You can use the ProofVerifier as is without subclassing it if you have the standard output variables which are the timestamp and the constraint status. If your proof function has custom output parameters, then you need to subclass this class and override the parseCustomOutput method.

```java
export default class ProofVerifier<Type {
  private zkComponentPath: string;
  private zkComponentName: string;

  constructor(
    zkComponentPath: string,
    zkComponentName: string,
  ) {
    this.zkComponentPath = zkComponentPath;
    this.zkComponentName = zkComponentName;
  }

  protected parseCustomOutput(customOutput: Array<string>): Type
  return null;

  verifyProof = async (customOutput: Array<string>): Type
  return null;
```
OZKi

Framework

Overview

OAuth

OAuth (Open Authorization) is the standard protocol for authentication and authorization used by many sites.

OAuth is used by many companies such as Google, PayPal, Facebook, etc., which allows users to share information on their PII with third-party applications or websites.

How is OAuth related to OZKi?

OZKi's design is orthogonal to OAuth. The PII provider can choose the authentication or access protocol it wishes to use, including OAuth, and the OZKi proof system works independently. The OZKi-BOT demo apps showcase the proof of payment with PayPal and proof of login with Google, both of which the PII providers use OAuth.

The high-level view of the OAuth is illustrated by this image (source: wikipedia):
Why OZKi?

**INNOVATIVE**
Unique proof tooling for webapp development.

**EASE OF USE**
Offers flexibility and simplified integration for web applications.

**SECURE PII RETRIEVAL**
Adds additional layers of security on top of the core ZKP system.

**USEFUL FEATURES**
Proof of Payment with PayPal, Proof of Login with Google, Proof of Key, etc.

**OPPORTUNITY**
Building block for Proof-As-a-Service business opportunities.

**PERFORMANT**
OZki processing typically completes under 2 seconds.
Next Steps

High-level proof function language

- Create an easy-to-use, high-level language which compiles into circom
- Further simplify OZKi interface

Proof-as-a-Service

- In its current form, OZKi is a developer’s tool and framework
  - However it is possible to use to extend the OZKi oracle as proof-as-a-service offering to the users

Open Source ZKP Community Involvement

- OZKi is an open source project
- We welcome the developer communities to improve OZKi, or fork it, and bring it to the next level
Thank you for listening! 

Feel free to contact us with any inquiries
https://vizualkei-labs.gitbook.io/ozki/project/forum

The Proof is in OZKi

For more information visit https://vizualkei-labs.gitbook.io/ozki/