Digital Assets Private Publication and Authorization on Blockchain using ZKP

Wei Mao， Lu Gan

Primas https://primas.io

Introduction

Zero-Knowledge Proof is one of the most promising research areas in the Blockchain industry. More recently, attention has been drawn to its potential in solving the 2 largest obstacles to mainstream adoption of Blockchain, extensibility and privacy.

As is well recognized, data recorded on Blockchain is open to the public. Everyone can read blocks and extract the data inside, which renders Blockchain unusable for a wide range of situations where data privacy is important. Take Bitcoin as an example in which the UTXO model is used to record token transfers. Anonymity is achieved by the separation of addresses from the real person but if the correspondence between one address and the person is exposed, with the help of some on-chain data analytics tools, one could easily find out all the related UTXOs of this person and calculate his total balance. The same thing happens for smart contract Blockchains such as Ethereum, where all the contract codes, all the invocations of the contracts and their parameters are visible to everyone.

ZKP could be used to solve those problems (we will not cover the details of how ZKP works in this paper though). With some well-designed solutions on top of ZKP, we could implement the same functions Blockchain provides whilst at the same time, keeping all the data private. ZCash, for example, implements the same functions as Bitcoin whilst keeping all the transaction data (the sender, receiver and amount) private using a transaction mixing mechanism which is implemented using zk-SNARK.

Blockchains do far more than just transfer tokens, however. During this revolution of information, most enterprises are experiencing a transition from manufacturing-driven to innovation-driven value, where an enterprise's most important assets are changing from money, materials and factories to talent, knowledge (information) and human networks, or in a professional term, 'social capital'. It is becoming more and more important for enterprises to protect their data assets and create the most value from them.

Blockchain makes it easy for digital assets to be circulated but is incapable of protecting them during that circulation. ERC-721 and ERC-1155 on Ethereum, standardized Non-Fungible Tokens; this has eased the process of creating and trading digital assets on Blockchain. This transparent circulation has meant that alongside ownership of digital assets, entire transaction history is also visible. For organizations, this means their sensitive information is being leaked to the public.

Earlier this year, Ernst & Young released a project called Nightfall, which enables the private transactions of ERC-721 tokens on Ethereum. After transferring the tokens into a contract, later transfers of the same tokens inside the contract are made invisible to the outside world. The creation of ERC-721 tokens, however, is still public. For data assets such as photos and articles, however, what commonly happens is the authorization of use is purchased, rather than the actual ownership of the asset. This is not supported by the NFT model, and there are no existing solutions to the privacy problem in this scenario.

The protocol we propose in this paper addresses this problem. The protocol supports the private registration of data assets, the private authorization of them using ZKP. In more detail, the functions of the protocol are:

1. Registration of digital assets on the Blockchain, without revealing information about the owner.

Using a photo as an example: One could register a photo on the Blockchain with only the id and hash of the photo made public. The photo's owner would not be identifiable.

If the owner wants to prove ownership when he publishing the photo, he could attach some text to the photo. The text could verify the ownership when querying on the Blockchain. Even with this text revealed, no one could find any other photos published by the owner on the Blockchain, nor could anyone create a fake proof of ownership.

2. Granting authorization of digital assets to users without revealing the buyer, the owner or any details of the digital asset.

When someone finds the photo online and wants to purchase it for their own use, they could get receive authorization from the Blockchain, without revealing the buyer or seller’ s information, or even the ID of the photo being purchased.

Again, if the buyer wants to prove using the photo was authorized, some text could be attached. Anyone could verify the validity of this authorization by sending this text to the Blockchain contract. The seller’ s information is kept private, and no other purchase records would be able to be found.

Although above has used a 'photo' as an example, this protocol could be used for any kind of digital asset.

We will explain the protocol in detail in the next chapters.

Currently we have already finished the implementation of this protocol on our own Blockchain platform and have been running tests for quite a while. In the future we intend to open source the code and publish it on Github.

The transfer of ownership is not covered in our protocol since there are already existing solutions for it. Our protocol could be easily integrated into existing solutions and we will provide a fully integrated solution in the open sourced codes.

In our current implementation, ZoKrates is used to develop the protocol, the ZKP algorithm used is Groth 16. The separation of ZKP algorithm and the upper level design allows the easy switch of ZKP algorithms. One could easily use Bellman to replace ZoKrates, or using a more advanced algorithm such as Bulletproofs and Sonic for better performance and the removal of trusted setup.

Variables

The variables used in the protocol are as follows:

Symbol	Description
A, B, C	Organization A, B, C
n_A	The registered name of organization A
h	sha256 hash function
$pk_A^E, addr_A$	The ECDSA public key and address of organization A
sk_A^E	The ECDSA private key and address of organization A
pk_A^R, sk_A^R	The public and private key of asset for organization A, where $pk_A^R = h(sk_A^R)$
α	A unique id representing some non-fungible asset
R_A^α	Organization A's registration record for the asset α
Z_B^α	A record that authorize the asset to B
π	The proof of zk-SNARKs

ϕ_L	The path from a leaf L to the root of a Merkle Tree
Ψ_L	The sister path from a leaf L to the root of a Merkle Tree
M	A calculation function for producing the Merkle root，the inputs are Ψ_L and L
MT^R	A Merkle tree to store organization's registration record
MT^Z	A Merkle tree to store asset's authorization record
σ	A random salt

Protocols

Organization Registration

Each organization has a public-private key pair representing the ownership of the assets, we call them "asset keys". Each of them is 32 bytes, recorded as pk^R, sk^R respectively, which satisfies:

$$pk^R = h(sk^R)$$

The organization also has a pair of public-private key representing their blockchain identity and the corresponding blockchain address, denoted as $pk^E, sk^E, addr$ respectively, generated using a standard elliptic curve digital signature algorithm (ECDSA).

The organization must carefully store the private keys in the two key pairs and should not expose them.

The organization also has a name n, which is provided to the querying party when the authorization information is verified, and should be formally a natural semantic information representing the identity of the organization.

The organization needs to complete the organization registration process according to the protocol before it has the authority to perform anonymous authorization of digital assets. Organization registration actually records some important information of the organization on the blockchain, and binds the organization's $pk^R, addr, n$ information together. Some of this information cannot be changed, such as $addr$ and pk^R; and other information requires permission management, such as the organization's registration name n.

We must guarantee the following points:

1. Only the organization that knows sk^R can complete the organize registration for the corresponding pk^R;

2. Once the organization has completed registration, only the address with the private key sk^E corresponding to pk^E can modify the organization's registration name.

Note that we have an implicit requirement for the above two points: the private key of the organization cannot be made public.

We rely on zero knowledge proof to achieve the first point; the implementation of the second point is based on the elliptic curve digital signature algorithm mechanism.

The following is an example of organization A, which specifically describes the process of organization registration:

1. Generate the required two sets of public-private key pairs and the other organizational information data in a secure environment;

2. Generate the zero knowledge proof π, including constraints:

   - $pk_A^R == h(sk_A^R)$

3. public inputs: $[pk_A^R]$

4. private inputs: $[sk_A^R]$

5. Call the method of smart contract:
   $Organization.register(\pi, pk_A^R, n_A, addr_A)$

Note: In the zero-knowledge proof generation process of ZoKrates, we can make a choice that whether the input variables are exposed, the public variables will appear in the generated proof, and the private ones will not appear.

Asset Registration

Organizations that have completed the registration are required to complete the registration process for their digital assets in accordance with the protocol in order to anonymously authorize these assets. For any digital asset, the organization needs to generate a unique numeric id for it, denoted as α. In this protocol, the numeric id can be any data that does not exceed 32 bytes.

Asset registration actually records the correspondence between the organization's asset public key pk^R and α on the blockchain, we must guarantee the following points:

1. Only organizations that know sk^R can register assets for the corresponding pk^R;

2. The same α can only be registered once.

Similarly, a zero-knowledge proof can help us achieve the first point.

We take the organization A and asset α as an example to introduce the process of asset registration:

1. Generate a random salt σ_A

2. Calculate $R_A^\alpha = h(\alpha / pk_A^R / \sigma_A)$

3. Generate zero knowledge proof π, including constraints:

   - $pk_A^R := h(sk_A^R)$

- $R_A^\alpha == h(\alpha / pk_A^R / \sigma_A)$

4. public inputs: $[R_A^\alpha, \alpha]$
5. private inputs: $[sk_A^R, \sigma_A]$
6. Call the method of smart contract: $Shield.register(\pi, R_A^\alpha, \alpha)$

Asset Authorization

A registered organization can anonymously authorize a registered asset to another registered organization. The essence of the authorization is to generate an authorization record on the blockchain, which implies the asset α and the information of the authorized party. We must guarantee the following points:

1. Only a registered asset α can be authorized;
2. The authorized party knows the private asset key sk^R corresponding to α;
3. The asset α and the information of the parties involved in the authorization cannot be made public.

Among the above-mentioned:

- The first point requires us to generate the zero-knowledge proof to prove the existence of the registration record for the asset α on the blockchain;
- The second point requires us to generate the zero-knowledge proof that the authorized party knows sk^R;
- As long as the above zero knowledge proof is given, the third point is naturally established.

The following takes the organization A authorized assets α to the organization B as an example to specifically introduce the process of asset authorization:

1. Get the pk_B^R;
2. Generate a random salt σ_{AB}, which will be sent to organization B through the encrypted channel;
3. Calculate $Z_B^\alpha = h(\alpha | pk_A^R | pk_B^R | \sigma_{AB})$;
4. Get $\psi_{R_A^\alpha}$, the sister path of R_A^α in MT^R from the **Shield** contract;
5. Get the latest $root_R$ of MT^R;
6. Generate zero knowledge proof π, including constraints:
   - $pk_A^R := h(sk_A^R)$
   - $R_A^\alpha := h(\alpha | pk_A^R | \sigma_A)$
   - $root_R == M(\psi_{R_A^\alpha}, R_A^\alpha)$
   - $Z_B^\alpha == h(\alpha | pk_A^R | pk_B^R | \sigma_{AB})$;
7. public inputs: $[Z_B^\alpha, root_R]$;
8. private inputs: $[pk_B^R, sk_A^R, \alpha, \psi_{R_A^\alpha}, \sigma_A, \sigma_{AB}]$;
9. Call the method of smart contract: $Shield.authorize(\pi, Z_B^\alpha, root_R)$.

Proof of Authorization

In order to show a third party that they have obtained authorization for a digital asset, the organization needs to generate a proof of authorization. The proof of authorization essentially proves the correspondence between the organization and the authorization record on the blockchain, that is:

1. There is indeed an authorization record Z^α corresponding to α and pk^R on the blockchain;

2. The organization owns the private key sk^R corresponding to pk^R.

The following takes the organization B and the asset α as an example, and specifically describes the process of generating the proof of authorization:

1. Get pk_A^R and σ_{AB}
2. Calculate $Z_B^\alpha = h(\alpha | pk_A^R | pk_B^R | \sigma_{AB})$;
3. Get $\psi_{Z_B^\alpha}$, the sister path of Z_B^α in MT^Z from the **Shield** contract;
4. Get the latest $root_Z$ of MT^Z;
5. Generate zero knowledge proof π, including constraints:
   - $pk_B^R == h(sk_B^R)$
   - $Z_B^\alpha = h(\alpha | pk_A^R | pk_B^R | \sigma_{AB})$
   - $root_Z == M(\psi_{Z_B^\alpha}, Z_B^\alpha)$
6. public inputs: $[pk_A^R, pk_B^R, \alpha, root_Z]$;
7. private inputs: $[sk_B^R, \psi_{Z_B^\alpha}, \sigma_{AB}]$;
8. $[\pi, pk_A^R, pk_B^R, \alpha, root_Z]$ is open to everyone, representing a proof that B has the authorization of asset α from A.

Verification of Authorization

For the proof of authorization provided by other organizations, the validity of the authorization certificate can be verified by calling the smart contract method:

$$Shield.approveCheck(\pi, pk_A^R, pk_B^R, \alpha, root_Z)$$

which returns**:**

1. Whether B is authorized by A;

2. The information of organization A；

3. The information of organization B.

Smart Contracts

The protocol uses a series of smart contracts，including the following categories：

- Organization related：used for organization registration，organization name modification，etc；
- Shield contract：used for registration，authorization and verification of assets；
- Verifier contract：Use the elliptic curve pairing function to verify the zero-knowledge proof as a library for other contract calls.

The following sections detail some of the important contracts：

Organization.sol

- $register(\pi, pk_A^R, n_A, addr_A)$:
  - Register the organization on the blockchain and record the correspondence between the organization's registration name，asset public key and ECDSA address.
- $resetName(n_A)$:
  - Modify the organization name，only the organization's registered ECDSA address has permission to call the method.
- $get(pk^R)$:
  - Obtain the organization registration name corresponding to the asset public key pk^R.

Shield.sol

- $register(\pi, R_A^\alpha, \alpha)$:
  - Register the digital asset α on the blockchain, where the information for organization A is hidden;
  - R_A^α will be inserted to $MT^R;
  - update MT^R;
  - generate a digital asset for α, which belongs to the Shield contract.
- $authorize(\pi, Z_B^\alpha, root_R)$:
  - Authorize the digital asset α to B, where the information for the digital asset α is hidden;
  - The contract needs to verify that the root was or are the root of MT^R.
- $verify(\pi, inputs)$:
  - Verify that an evidence π matches the corresponding inputs by calling the Verifier contract method.
- $authorizeCheck(\pi, pk_A^R, pk_B^R, \alpha, root_Z)$
  - Verify the authorization information.

Reference

[1]EY Global Blockchain R&D. Nightfall: Zokrates library for private token transfer over the ethereum blockchain.

[2]Stefan Deml Jacob Eberhardt, Thibaut Schaeffer. ZoKrates.

[3]Vitalik Buterin. zk-SNARKS: Under the hood.

[4]Zcash. How Transactions Between Shielded Addresses Work?

[5]Zcash. What are zk-SNARKs?

[6]Jens Groth. On the Size of Pairing-based Non-interactive Arguments.

[7]Bellman

[8]Benedikt B¨unz. Jonathan Bootle. Bulletproofs: Short Proofs for Confidential Transactions and More

[9]Mary Maller. Sean Bowe. Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings