zkFAITH: Soonami's Zero-Knowledge Identification and Authentication Protocol

Author: Mina Namazi Concept: Duncan Ross

September 15, 2022

Abstract

Privacy-preserving identification systems that enable individuals to prove their eligibility for using specific services are a challenging topic. Cryptographic techniques are deployed to construct proofs of identity across the internet. However, they do not prevent users from forging and submitting fake data. We design, implement, and evaluate a protocol called "zkFAITH" in this paper. The zkFAITH is a new approach introduced in this paper to obtain a verified zero-knowledge identity unique to each individual. The protocol verifies the identity of the individuals and issues the identity without revealing any information to the authenticator or verifier by leveraging a zero-knowledge proof system. After an authority evaluates the information's correnctness and confirms it by making a tag, we deploy a CL signature scheme to create a statement in addition to a zero-knowledge proof. The users can deploy their zkFAITH to access various services. In this paper, we show the design and implementation of the zkFAITH with the generated proofs in real-world scenarios and discuss its scalability.

1 Introduction

A blockchain is a distributed computation record between many devices known as nodes with no trusted setup. All the nodes access the updated data, run the required calculations locally, combine them, and provide the globally agreed, immutable history. However, the parties' identity and transmitted data might be publicly available. This widespread availability of individual data was not desirable to FinTech. Hence, privacy-preserving distributed systems received significant attention. Solutions such as mixing, decoys in tokens such as Monero, and systems based on zkSNARKs, such as Zcash [5], are introduced to protect sensitive information of individuals.

The expected privacy in finance is not limited to the blockchain space. Anonymous identification and authentication is another zone the privacy of the users plays a crucial role. It is common for group members to log in to a service without revealing their identity. Hiding the transaction history is required in some scenarios when users are willing to take actions anonymously, such as voting without linking to each other. Moreover, individuals are required to prove their eligibility to access various organizations' services. For example, a streaming platform requires a user's age to be above 18 and their location in Canada. If users desire to register on this platform and access the content, they need to show their ID card information, to prove their age and location. A problem appears when the id card contains other information such as nationality, height, eye color, and religion that the streaming platform does not need to see or that the user is unwilling to reveal to the streaming platform. Therefore, a universal identification protocol is required to allow people to prove they are legitimate users without revealing extra information.

In decentralized platforms, stakeholders deploy cryptographic methods to provide privacy-preserving identification and authentication mechanism. Practical computational integrity proof systems, like SNARKs, STARKs, and Bulletproofs, found numerous potential use cases in the recent investigation. These proofs are required not to reveal any additional information beyond the statement of the prover; hence they are called zero-knowledge (ZK). In ZK scheme, a prover, holds a statement willing to convince the other party, verifier, of its correctness without revealing any further information. Often, these proof systems require expensive computations. When ZK proofs are deployed as a verification system, the proof size must remain small regardless of the computation complexity.

Additional to scalability challenges of ZK proofs in real-world scenarios, they have no built-in mechanism to prevent individuals from submitting fake data. Current techniques cannot prevent if a malicious user who is not eligible to access a streaming platform use the id card information of another eligible user to gain access.

Our contribution. This work focuses on developing a zero-knowledge identification method with data integrity in decentralized platforms called ”zkFAITH.” The proposal combines various phases working as a black box without any trusted authority. First, it allows users to build a credential that perfectly hides their essential information while using it. The information inside the credential gets authentication from government-approved authorities. After the credential issuance, the user can prove to possess a valid credential with zero knowledge. Showing the zkFAITH leaks no data beyond the statement the user is willing to prove. Later, to access the organization’s services or perform any financial transactions, zkFAITH can be attached to the request/transaction. The verifier accepts the request/transaction only if convinced that the data inside the request/transaction belongs to the zkFAITH’s owner and the identity is legitimately issued. The proposed protocol provides a scalable solution to protect the privacy of individuals and provide data integrity simultaneously.

The rest of this paper is structured as follows. In Section 2, we comprehensively review recent developments in blockchain in which zero-knowledge proofs are used and implemented. In section 3, we describe our proposal of zkFAITH identity after explaining the core cryptographic schemes to build it. Section 5 concludes the discussion of the proposed solution and represents open lines of future work.

2 Related Work

This section describes the previous work that has been done in literature to achieve an efficient zero-knowledge identity. We discuss the advantages and disadvantages of their work and illustrate the differences between our solutions.

2.1 Deco

DECO [1], is a decentralized oracle for TLS, which is source-agnostic and supports any website running standard TLS with no server-side cooperation. Similar to our scenario, a prover commits to a piece of personal and private data and proves to a verifier that this data is obtained from a TLS server. It also generates proof of knowledge of the data. For example, in the proving age scenario, the proof is the predicate "$y/m/d$ is the prover's birth date and the current date - which should be at least 18 years." DECO authenticates the provided information of the prover. The verifier must be convinced that the asserted proof about the data is accurate and that this data is certainly obtained from the website. The protocol is privacy-preserving since the verifier only observes the provided proofs about the data and checks their validity. These proofs leak no information to the verifier about the data or the prover.

The protocol comprises a three-party handshake phase to establish session keys, a query execution phase where the prover fetches the server for data, a proof generation phase where the prover proves the query is well-formed and the response satisfies the desired condition.

Deco claims to solve the problem of authenticity in zero-knowledge credential protocols. However, the three-party handshake and the deployed secret-sharing schemes seem expensive operations. We achieve authenticity in our proposed solution with no TLS and handshake requirements. In our scenario, an authenticator is a government-approved entity with no access to the parameters to execute the protocol and solely provides authenticity.

2.2 zh-cred

A similar approach to our proposed solution is the zk-cred system introduced in [2]. Usually, the digital platform's users must prove they are legitimate to access the platform. They prove they are human or located in a specific place to access the local services. The zk-cred avoids making unrealistic assumptions where multiple trusted sources exist to issue credentials. It removes the burden of the credential issuers to store various signing keys and proposes a solution using the general zero-knowledge proof system.

A credential in zk-creds is a commitment to arbitrary attributes, such as the fields name or birth date from a passport, placed on a list. The users must convince the credential issuer about possessing it. They must keep a witness to the credential's membership in the issued list and ask the issuer for an updated witness. They deployed a Merkle tree approach to save the list. Therefore, just updating one witness element in the tree is straightforward. The issuer maintains a list of publically verifiable credentials. Anyone can access the list and verify the issue process. Later, users desire to show their credentials and prove some criteria, such as age. They produce zero-knowledge proof that their credential is listed in a Merkle tree of all issued credentials and meets the access criteria, which is publically verifiable. The revocation mechanism is simply removing the credential from the list.

A credential "cred" is a commitment to some attributes matching the digital identity documents with some random information. They use the Pederson commitment scheme. There is an issuer who keeps the list of credentials in a list as a Merkle tree. The user sends cred to the issuer with supporting documents such as a zero-knowledge proof or a digital signature to convince the issuer of specific criteria. If the issuer is satisfied, it adds the cred to its list and returns the Merkle authentication path.

Although the zk-cred approach is efficient, in real-world scenarios, it is unrealistic to adopt it by the issuers. Therefore, as the main difference with our proposed solution, we assume this entity has no access to the zero-knowledge-proof system's public parameters and cannot issue or verify any signature/proof. Moreover, our scheme provides data integrity before issuing any ZK identity to prevent obtaining an id on submitting fake information. Hence, in our scenario, an extra trusted party approved by the government controls the integrity of the submitted information. The identity issuance happens by an unbounded DAO upon receiving an authentication tag from this entity.

3 zkFAITH

This section describes our proposal for a zero-knowledge identification and authentication system of "zkFAITH". First, we describe the core building blocks to construct our solution. Then, we explain the zkFAITH protocol and its deployment in real-life scenarios. Informally, the parties are issued a zero-knowledge identity after a third-party government-approved authority confirms their information. Each party can obtain various identities on their different legal documents. However, a unique identification number, such as the passport number of each party, is a shared value in all identities. The parties can prove that they own a legitimate identity based on their information with zero knowledge. These data are updatable. Therefore, if a passport is expired, the user can ask to update their zero-knowledge identity. The zkFAITH is unique, and there is no way that a user is issued two different identities for the same information. In case of malicious activity, such as sharing the credential, the deployed revocation mechanism removes the issued credential and is no longer functional. We explain the details of the protocol in the following sections.

3.1 Preliminaries

This section provides the core cryptographic primitives required to develop our zk-FAITH proposal.

3.1.1 Notation.

We denote sampling uniformly from a set S by $y \leftarrow S$. Proof of knowledge of a relation $R = \{(x; w) : P(x, w)\}$ for an instance x is a proof of knowledge of the witness w such that $P(x, w)$ is satisfied, for a predicate P. A commitment to a value x with randomness r is $Com(x; r)$. The calligraphic letter $\mathcal{X}$ denotes the parties playing in the protocol. Value x inside brace $[x]$ shows the encrypted version of x.

3.1.2 Non-Interactive Zero-Knowledge Proof of Knowledge (NIZK)

Groth [3] introduced a proof system with the following functionalities.

- $\mathrm{NIZK.Setup}(1^\lambda) \to pp$: generates public parameters for the bilinear group. It is the input to all other algorithms.
- $\mathrm{NIZK.Prove}(x, w) \to \pi$: generates a proof, where x is the statement and w is the witness.
- $\mathrm{NIZK.Verify}(\pi, x) \to \{0, 1\}$: verifies the proof π with respect to x and returns 1 as a truthy value.

3.1.3 CL Signatures

A digital signature scheme is a way of signing documents and a functional building block to construct an anonymous identification protocol. In this paper, to build an efficient identity-based solution for individuals, they must get a signature on their confidential information. Hence, we deploy Camenisch-Lysyanskaya (CL) signature scheme [4] to obtain the signature. Later, we explain how to prove the knowledge of this signature on their information in a zero-knowledge way.

$\mathrm{CL.Setup}$: inputs the security parameter 1^λ and outputs $pp_{CL} = (\mathbb{G}, G, \mathbf{g}, g, e)$. These parameters are inputs of all the other algorithms.

$\mathrm{CL.KeyGen}$: each user does the following

- Choose $x \leftarrow Z_q, y \leftarrow Z_q$, and for $1 \leq i \leq l, z_i \leftarrow Z_q$.
- Let $X = g^x, Y = g^y$ and, for $1 \leq i \leq l, Z_i = g^{z_i}, W_i = Y^{z_i}$.
- Return $sk = (x, y, z_1, \ldots, z_l)$, and $pk = (q, \mathbb{G}, G, \mathbf{g}, g, e, X, Y, \{Z_i\}, \{W_i\})$.

$\mathrm{CL.AskSig}$: the algorithm inputs the committed message $M = g^{m^{(0)}} \Pi_{i=1}^{l} Z_i^{m^{(i)}}$, where M is commitment to a set of messages $(m^{(0)}, m^{(1)}, \ldots, m^{(l)})$, signer's secret key $sk = (x, y, z_1, \ldots, z_l)$, and public key $pk = (q, \mathbb{G}, G, g, \mathbf{g}, e, X, Y, \{Z_i\}, \{W_i\})$. Then it continues as follows.

- The user sends a proof of knowledge of the commitment's opening to the signature issuer.
$$PK\{(u^{(0)}, \ldots, u^{(l)}) : M = g^{u^{(0)}} \Pi_{i=1}^{l} Z_i^{u^{(i)}}\}$$

$\mathrm{CL.IssueSig}$: The signature issuer will run the algorithm, and acts as follows if satisfied with the proof of knowledge of the commitment opening.

- The issuer chooses a random $\alpha \to \mathbb{Z}_q$, calculates $a = g^\alpha$. Let $A_i = a^{z_i}$, for $1 \leq i \leq l$, let $b = a^y, B_i = (A_i)^y$. Let $c = a^x M^{\alpha x y}$.
- The user outputs the signature as $\sigma = (a, \{A_i\}, b, \{B_i\}, c)$.

$\mathrm{CL.Verify}$: the algorithm is run by the user to verify the correctness of the signature. It inputs $pk = (q, \mathbb{G}, G, g, \mathbf{g}, e, X, Y, Z_i)$, message M, and signature $\sigma = (a, A_i, b, B_i, c)$, and checks as it follows.

- $\{A_i\}$ were formed correctly: $e(a, Z_i) = e(g, A_i)$.

- b and $\{B_i\}$ were formed correctly: $e(a, Y) = e(g, b)$ and $e(A_i, Y) = e(g, B_i)$.
- c was formed correctly: $e(X, a) \cdot e(X, b)^{m^{(0)}} \Pi_{i=1}^{l} e(X, B_i)^{m^{(i)}} = e(g, c)$.

This signature scheme is a secure two-party computation of a signature on a discrete logarithm representation of the message M under the signer's public key. The signer can only see a commitment to an array of messages where the commitment scheme is hiding and does not reveal any information about the raw data of the user to the signer.

3.2 System Model

The zkFAITH identity solution comprises a Claimant $\mathcal{C}$ who wishes to generate a zero-knowledge identity proof via a commitment to various claims with supporting documents (i.e., date of birth, country of residence, complete passport information, driving license, or medical certificate). This information is signed with their private key and generates proof of knowledge of the data in the claim. An Authority $\mathcal{A}$ is a government-approved organization that can verify the authenticity and integrity of the data submitted by $\mathcal{C}$. If $\mathcal{A}$ is convinced that the proofs generated by the $\mathcal{C}$ are valid proof that the commitment information belongs to the Claimant, it adds an authentication tag on the submitted data that assists in constructing the zkFAITH identity.

There is a verifier $\mathcal{V}$, in our scenario Soonami (or Unbounded DAO), who receives the authentication tag from the $\mathcal{A}$ and adds it to the commitment/proof of $\mathcal{C}$, and issues the zero-knowledge identity of zkFAITH to $\mathcal{C}$.

A Compliance Officer $\mathcal{O}$ is a third-party regulatory trusted organization wishing to trace transactions or activity of potentially malicious claimant (owner of particular zkFAITH identity). Furthermore, a Compliance DAO $\mathcal{D}$ is a decentralized organization whose sole purpose is to govern the compliance access to zkFAITH of the Compliance Officer via a trusted voting mechanism.

The zkFAITH is a protocol that aims to create compliant data proof for identity, medical data, rights, membership, and other forms of personal data stored in a secure wallet with on-chain verification and data integrity check by a recognized authority, therefore providing real-world compliance, a form of KYC and usability without compromising or revealing user data.

Each identity is not self-sovereign and can not issue a claim on another identity (person, organization, or system/machine), preserving compliance and verifiability of the claimed data. The protocol is zero-knowledge; therefore, the identity of the parties and transmitted data reveal no information about the parties or the transactions.

The $\mathcal{C}$ starts the protocol by sending their documents and public key with a commitment to them, and wallet id number, to the authority $\mathcal{A}$, who checks their validity. If $\mathcal{A}$ is satisfied with the correctness of the received information, it generates a tag and sends the same committed values to $\mathcal{V}$. Then, $\mathcal{C}$ sign (commit to) to the information and send the commitment and proof of an opening to the $\mathcal{V}$. The verifier signs the claimant's commitment with their private key. Then they generate proof of knowledge of the signature and send them to the claimant; a unique zero-knowledge identity for each user.

This procedure is a one-time operation. Later, suppose the users desire to prove their eligibility for accessing a specific web service in which the only requirement is proving the user is above 18 years old. In that case, it suffices to generate a subproof of knowledge of the user's birth year and attach it to the zkFAITH identity. Showing the zkFAITH identity along with this subproof can convince the website to grant access to the user and allow them to benefit from their services. We explain the interactions between parties in Figure 1.

3.3 Security Goal

The proposed solution's security goal is to protect the privacy of individuals and provide data integrity. Privacy means that in any step of the protocol, any adversarial behavior cannot bind the result of the protocol interactions to any specific claimant. Various deployments of the zkFAITH Id are unlinkable to each other and reveals nobody's identity. We assume that the authenticator, $\mathcal{A}$, is a trusted party with no access to the system parameters, and no information storage related to the individuals after authenticating them is permitted.

The authority $\mathcal{A}$ is a trusted party. However, we assume that it only controls the validity of the provided document and does not store them. In our scenario, the claimant C can be controlled by malicious adversaries who are allowed to alter the protocol instructions and seek to modify the inputs to corrupt the system are among instances that our security goals address. On the other side, the verifier is semi-honest who follows the instructions of the protocols and cannot modify the data. However, it can be curious and can get unauthorized information from the interactions and outputs of the protocol. The authenticator and verifier in real life scenarios are not colluding because such a colliding will endanger the reputation of the both organisations.

3.4 Design Challenges

This section lists potential attack scenarios a malicious adversary may deem to attempt in our system model. We also outline our proposed solution for each scenario. While the attacks are not confined to the listed, we emphasize the design thinking that would overcome the security challenges by determining an intuition of the cryptographic primitives.

One possible attack scenario is when a malicious user (hereafter known as "attacker") generates fake profiles using the identification information of other honest users (hereafter known as "claimant"). Let us assume that a claimant obtains a zkFAITH identity on their passport. The attacker, who owns no driving license, takes control of the claimant's document and submits it to ask the verifier for a zkFAITH on the driving license by committing to their driving license information and the malicious user's public key. If the verifier is convinced with the claimant's proofs, it issues the zkFAITH for the malicious user. We defeat this attack by requiring an authentication step before issuing the zkFAITH. The claimants and the validity of the represented document should be authenticated, and after a tag is generated and sent to the verifier, it continues the protocol. Otherwise, no new zkFAITH for the provided driving license is generated, and the protocol halts.

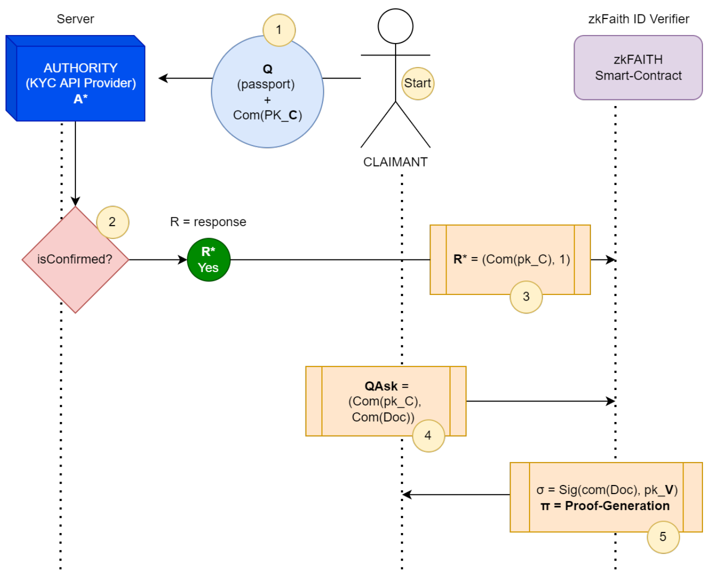

Figure 1: Interactions of the parties

Another possible scenario is that the verifier issues an identity on arbitrary messages not provided by the claimant and uses the claimant to access some services through them. The protocol prevents the attack by requiring the claimant to check the issued signature by the verifier for the ”correct computation” on the provided data. The protocol aborts if the generated CL signature by the verifier does not satisfy the verification step of the claimant.

Another scenario is when an organization with some criteria to allow the users to access their services requires the claimants to prove that they meet their required criteria. They can relate two shown credentials to each other and trace them back to identify the individuals. We defeat this attack by deploying a zero-knowledge-proof system that is randomizable. Various shows of the zkFAITH on the same document or two zkFAITH issued for different documents of the same claimant are not linkable.

3.5 zkFAITH Construction

This section combines all the preliminaries and constructs a zkFAITH protocol. The zkFAITH protocol comprises of $Faith = \{\text{F.Setup}, \text{F.KeyGen}, \text{F.Auth}, \text{F.Ask}, \text{F.Issue}, \text{F.Show}, \text{F.Revoke}\}$. The definition of each step is explained as follows. We assume that each user is pre-issued an identity document by the government. They receive a document doc with inf_i to be each section of the doc kept as a secret. Moreover, each user posses a unique wallet id, wid.

Set up and Key Generation:

- $pp_F \leftarrow \text{F.Setup}(1^\lambda)$: The algorithm takes a security parameter λ, calls the

CL.Setup and outputs the system parameters pp_{CL}. It assigns $pp_F = pp_{CL}$ and inputs them to all of the below algorithms.

An automated smart contract is required to view the public parameters, which can be implemented in the zkFAITH smart contract. The initial input values are passed at the time of deployment to the contract constructor. It is visible by everyone in form of a pure function or a constant variable. This step eliminates the trusted setup requirement.

- $(sk_i, pk_i) \leftarrow \text{F.KeyGen}(pp_F)$: Each party on the input pp_F calls the key generation algorithm of the CL signature CL.KeyGen locally to generate a key pairs for each user i.

Request:

- $R \leftarrow \text{F.Auth}(wid, doc, Com(pk_{\mathcal{C}}), Com(doc))$: $\mathcal{C}$ sends previously government-issued $doc = \{inf_1, inf_2, \ldots, inf_l\}$ with a commitment to its value, $Com(doc)$, the wallet id, wid, and a commitment to their public key $Com(pk_{\mathcal{C}})$ to the authority. We assume that $\mathcal{A}$ deploys a mechanism such as face recognition to control that the represented information is correct and belongs to the $\mathcal{C}$. Then, $\mathcal{A}$ outputs a response $R = (wid, Com(pk_{\mathcal{C}}, v), Com(doc; r), 1)$ to $\mathcal{V}$. Otherwise, it aborts the protocol.

- $Q \leftarrow \text{F.Ask}(Com(doc; r), Com(pk_{\mathcal{C}}, v))$: $\mathcal{C}$ takes the commitments to the information of the claimant's doc and public key. It calls NIZK.Prove to generate a proof of knowledge of each inf_i inside the doc and outputs π_{doc} on the given identity information. Moreover, it repeats the same procedure for public key and outputs a proof of corresponding secret key π_{pk}. The $\mathcal{C}$ sends $Q = (Com(pk_{\mathcal{C}}; v), \pi_{pk}), (Com(doc; r), \pi_{doc}))$ to the verifier $\mathcal{V}$ using the wallet address that matches the wid from the previous step. In this scenario, Soonami is the verifier who issues the zero-knowledge identity to the claimant.

Issue:

- $\sigma \leftarrow \text{Issue}(R, Q)$: The verifier inputs the authentication response from $\mathcal{A}$ for the corresponding wallet id of the claimant excluding is wid and 1 parameters. It also receives the request Q from the $\mathcal{C}$. Then, $\mathcal{V}$ compares the commitments from R and Q to be the same. Then, V calls $\text{NIZK.Verify}(\pi_{doc}, \pi_{pk})$ and if the output is 1, then verifier calls the CL.IssueSig on the $Com(doc; r), Com(pk_{\mathcal{C}}, v)$, and the wid, outputs σ.

  Moreover, the verifier encrypts the request $[Q] = Enc(Q, pk_{\mathcal{O}})$ and $[\sigma] = Enc(\sigma, pk_{\mathcal{O}})$ of the claimant σ under public key of $\mathcal{O}$. It adds wid and sends $E = (wid, [Q], [\sigma])$ to the compliance officer. This values are stored in a list by the compliance officer.

  $\mathcal{C}$ calls the signature verification algorithm $\text{CL.Verify}(\sigma, pk_{\mathcal{C}}, doc, wid) \rightarrow 0/1$. It takes as inputs the signature σ, public key of the claimant, and the inf_i. If $\mathcal{C}$ is convinced that the signature is correctly generated on the previously provided data, it outputs 1, and 0 otherwise.

The claimant generates a commitment to this signature and a proof of knowledge of the signature. It outputs zkFAITH= $(Com(\sigma; u), \pi_\sigma)$ is the unique zero-knowledge identity of the user for the provided document of the claimant.

Show:

We assume that there is an organization $\mathcal{M}$ with some defined criteria to provide access to its services. As an example, validity of claimant's visa date is required at a hotel reception to accept the $\mathcal{C}$ as guest. We call this criteria μ and $\mathcal{C}$ is required to prove that the previously issued zkFAITH, contains a valid visa expiry date that meets the $\mathcal{M}$'s requirement. $\mathcal{C}$ acts as it follows.

- The claimant calls the proof generation algorithm of the NIZK proof system and generates the proof that inf_i inside the doc meets the access requirement μ. It runs $\text{NIZK.Prove(zkFAITH, } \mu)$ and sends the proof π_μ to the $\mathcal{M}$.
- The claimant also needs to prove of possession of a valid zkFAITH. i.e., a membership proof to show that the zkFAITH exists in the $\mathcal{O}$'s list L. It calls $\text{NIZK.Prove}(wid, L, zkFAITH)$ and generates π_L.
- The claimant outputs (π_μ, π_L) and shows it to the $\mathcal{M}$ as a proof of knowledge of the correct criteria and proof of knowledge of a correct credential from a verifier.

Revoke:

- In case of fraud or change of any information in the previously issued document by the government, with the vote of $\mathcal{D}$, the compliance officer $\mathcal{O}$ decrypts E, and revokes the signature issued for the specified wid and deletes it from the list L.

The above protocol protects privacy of the individuals when they request a zero-knowledge universal identity. Moreover, data integrity and authenticiy is also provided by deployment of this solution. We extend Figure 1 to illustrate the deployments of the algorithm in the presence of a smart contract in Figure 2.

4 Discussion

The protocol introduced in section 3 perfectly protects individuals' identities in decentralized platforms. The authority $\mathcal{A}$ is a trusted party. However, it has no access to the common reference string generated during the protocol and parameters of the zero-knowledge protocol. Moreover, it will not store any information related to the parties. The "request" protocol leaks no information about the identity or raw data of the claimant since the commitment scheme is perfectly hiding. In real-world scenarios, the verifier is an automated smart contract. Its procedures are pre-defined. Hence there is no room for collusion between authority and the verifier contract. Later on, when the zkFAITH is deployed to access various services during the "show" phase, it leaks no information to the service provider. Since the identity and generated proofs are randomizable, different usage of it is unlikable to each other. The service providers cannot link two shown zkFAITH's of the same claimant to each other and reveal the claimant's

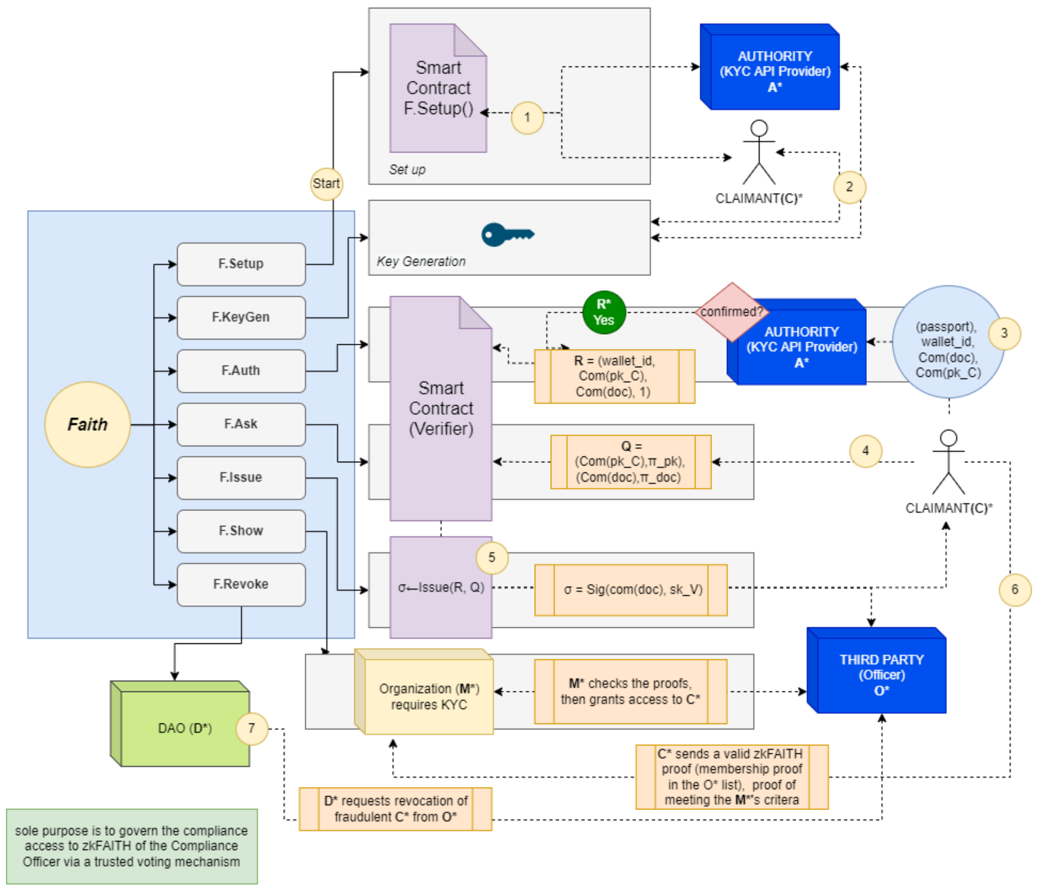
Smart Contract F.Setup()
1
Set up
AUTHORITY (KYC API Provider) A*
CLAIMANT(C)*
2
Start
Key Generation
F.Setup
F.KeyGen
F.Auth
F.Ask
F.Issue
F.Show
F.Revoke
Faith
R* Yes
confirmed?
AUTHORITY (KYC API Provider) A*
(passport), wallet_id, Com(doc), Com(pk_C)
3
R = (wallet_id, Com(pk_C), Com(doc), 1)
Smart Contract (Verifier)
Q = (Com(pk_C),π_pk), (Com(doc),π_doc)
4
CLAIMANT(C)*
σ←Issue(R, Q)
5
σ = Sig(com(doc), sk_V)
6
THIRD PARTY (Officer) O*
Organization (M*) requires KYC
M* checks the proofs, then grants access to C*
DAO (D*)
7
C* sends a valid zkFAITH proof (membership proof in the O* list), proof of meeting the M*'s critera
D* requests revocation of fraudulent C* from O*
sole purpose is to govern the compliance access to zkFAITH of the Compliance Officer via a trusted voting mechanism

Figure 2: Full Construction of zkFAITH in the Presence of Smart Contract.

identity. The relevant authority must have issued the data (i.e., Passport Office, Police, Vehicle Agency, Medical board). Hence, data schema and its format are known. Using CL signatures guarantees data integrity since the data structure is preserved in the process of signature/proof generation.

5 Conclusion and Future Work

We designed a zero-knowledge identification protocol, zkFAITH, and developed its verification procedure. We showed that the devised solution could be used as an anonymous credential to access various organizations' services with specific criteria. We proved that the zkFAITH solution is practical in real-world scenarios. It is a privacy-preserving protocol that simultaneously provides integrity and authenticity of the data. We showed with no trusted setup how our solution could be implemented in decentralized platforms and work with smart contracts with minimum gas fees.

As the second part of the protocol for future work, we will define compliance. Each user can create a transaction deploying the zkFAITH protocol and prove its correctness and eligibility for making the transaction with zero knowledge to a potential provider. The transactions and the proofs perfectly hide the origin, destination, and values inside the transactions against a malicious verifier.

Moreover, an ”update” protocol will be added to the scheme for scenarios where some parts of the previously issued document are changed, such as expiry dates. The protocol dynamically updates the altered information inside the zkFAITH with no requirement of initializing the entire system from scratch by deploying malleable signature schemes.

Finally, to increase the level of security and privacy in the scheme, an SDK will be embedded in the authentication part to allow the authenticator access to the system parameters and issue/verify commitments or signatures.

References

[1] Fan Zhang, Deepak Maram, Harjasleen Malvai, Steven Goldfeder, and Ari Juels. Deco: Liberating web data using decentralized oracles for tls. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1919–1938, 2020.

[2] Michael Rosenberg, Jacob White, Christina Garman, and Ian Miers. $\mathrm{z}k - creds$: Flexible anonymous credentials from zksnarks and existing identity infrastructure. *Cryptology ePrint Archive*, 2022.

[3] Jens Groth. On the size of pairing-based non-interactive arguments. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 305–326. Springer, 2016.

[4] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Annual international cryptology conference*, pages 56–72. Springer, 2004.