Trust Me If You Can: Trusted Transformation Between (JSON) Schemas to Support Global Authentication of Education Credentials

Stefan More, Peter Grassberger, Felix Hörandner, Andreas Abraham, Lukas Daniel Klausner

Trust Me If You Can: Trusted Transformation Between (JSON) Schemas to Support Global Authentication of Education Credentials*

Stefan More[1 [0000−0001−7076−7563]], Peter Grassberger[1,2], Felix Hörandner[1 [0000−0001−8591−3463]], Andreas Abraham[1 [0000−0002−4163−9113]], and Lukas Daniel Klausner[3 [0000−0003−3650−9733]]

[1] Graz University of Technology, Austria
{smore,fhoerandner,aabraham}@iaik.tugraz.at
[2] lab10 collective, Graz, Austria
p.grassberger@student.tugraz.at
[3] St. Pölten University of Applied Sciences, Austria
mail@l17r.eu

Abstract. Recruiters and institutions around the world struggle with the verification of diplomas issued in a diverse and global education setting. Firstly, it is a nontrivial problem to identify bogus institutions selling education credentials. While institutions are often accredited by qualified authorities on a regional level, there is no global authority fulfilling this task. Secondly, many different data schemas are used to encode education credentials, which represents a considerable challenge to automated processing. Consequently, significant manual effort is required to verify credentials.

In this paper, we tackle these challenges by introducing a decentralized and open system to automatically verify the legitimacy of issuers and interpret credentials in unknown schemas. We do so by enabling participants to publish transformation information, which enables verifiers to transform credentials into their preferred schema. Due to the lack of a global root of trust, we utilize a distributed ledger to build a decentralized web of trust, which verifiers can query to gather information on the trustworthiness of issuing institutions and to establish trust in transformation information. Going beyond diploma fraud, our system can be generalized to tackle the generalized problem for other domains lacking a root of trust and agreements on data schemas.

Keywords: Blockchain · Distributed Ledger · Web of Trust · Trust Management · Education Credentials · Verification · Self-Sovereign Identity

* This work was supported by the European Union's Horizon 2020 Framework Programme for Research and Innovation under grant agreement № 871473 (KRAKEN) as well as the Josef Ressel Center for Blockchain Technologies and Security Management (JRC Blockchains). The authors would also like to thank TU Graz' Registrar's Office for insights into verification of (paper-based) diplomas.

1 Introduction

When applying for a job or a study program, applicants need to provide evidence of their past education achievements (e. g. graduation diplomas) so that the human resources department or institution is able to assess the applicants' qualifications and eligibility. Because applicants occasionally create fake education credentials or modify credentials of legitimate issuers [7], universities spend considerable effort to verify the authenticity of documents. Nowadays, education credentials can be represented as signed digital documents [18], e. g. as Verifiable Credentials in the Self-Sovereign Identity (SSI) context [41]. While digital signatures simplify the verification process, two challenges remain, which we tackle in this work – namely to evaluate a credential's issuer's legitimacy as well as to interpret the various data formats of credentials (cf. Section 4).

Challenge 1: Because education diplomas provide substantial value, a whole industry of fake universities (so-called diploma mills [5]) has arisen, which exist for the sole purpose of selling fake diplomas. Within the UK alone, 240 universities were identified as such fake universities according to the UK's Department of Education [40]. To avoid accepting such fake credentials, a verifier needs to spend considerable effort, money and resources to perform authorization of credential issuers, which is especially complex in a global setting with many different types of issuers alongside universities.

Challenge 2: Besides verifying an education credential's authenticity and legitimacy, a verifier also needs to be able to interpret the credential's content. Various data formats have been proposed for digital education credentials [41,10], but so far, no unified, widely accepted standards have emerged [10]. Therefore, verifiers have to spend additional resources to interpret the content of education credentials manually or with external help, which also hinders automated processing.

Contribution 1: *Verify Legitimacy of Diverse Credentials and Issuers.* In this paper, we use a distributed ledger (DL) to enable all involved institutions to publish trust information about credential issuers. This effectively forms a web of trust (WoT), a directed graph of trust statements published by institutions and based on previous manual evaluations (cf. Section 5). This web of trust further enables verifiers to define their own trust policies on how to automatically evaluate a (previously unknown) institution's legitimacy.

Contribution 2: *Verify Credentials Issued in Different Schemas.* Building on the above web of trust, we introduce a second graph in which involved parties can publish, discover and authenticate information to transform credentials between schemas (cf. Section 6). Given this information, verifiers are able to locally transform a credential from the issuer's schema into their preferred schema, possibly via several intermediate schemas. Combined with our web of trust, verifiers can interpret a credential's content and assess the trustworthiness of transformation information.

Since the tackled challenges are not limited to education credentials, but also apply in other contexts, we point out that our approach can also be applied in other domains where no globally trusted authority exists.

We demonstrate the feasibility of our framework with an **implementation**, which uses a Solidity smart contract on an Ethereum-based DL to manage the web of trust and the distributed file system IPFS to store the transformation information (cf. Section 7). We also evaluate and discuss several properties of our system (cf. Section 8).

2 Background

In this section, we introduce concepts relevant for our approach. A **Distributed Ledger (DL)** is a redundant data store on distributed nodes without central control. The nodes agree on a state by running a consensus protocol, and such nodes can be run by various parties, such as organizations or companies. DLs can be accessed through different access models, depending on who is allowed to participate in the network by joining it (public vs. private) or by who has read and write access to it (permissioned vs. permissionless). Many ledgers, such as Ethereum, support **Smart Contracts (SCs)**. SCs are code stored on the DL that is deterministically executed by the nodes performing the consensus protocol. Variables in the code are stored on the DL, as well, and can be read and modified using functions supplied by the contract. When a user sends a function call to a contract, the resulting state is only written to the ledger if all nodes agree on the result of the computation. In contrast to centralized and federated identity models [46], **Self-Sovereign Identity (SSI)** gives control over the identity solely to the user [32] without having a central trusted party. In this model, users create their own identity by using a **Decentralized Identifier (DID)** [35]. This DID can be registered on a DL and is used to resolve key material and other information needed to interact with its subject. In addition, it is used to link statements about the user to their identity. **Verifiable Credentials (VCs)** are used as a typical data format in SSI systems and are specified by the W3C [38]. While the schema of the envelope is defined by the W3C, the claims inside a VC are encoded according to a JSON schema [44] so that credentials are sufficiently flexible for various use cases. Obviously, this flexibility of the content's schema also represents a challenge for verifiers when facing content in an unknown schema. The **InterPlanetary File System (IPFS)** [34] is a distributed file system based on an open peer-to-peer network. Similar to a public DL, there is no central party controlling the system and everyone can join the network of nodes and retrieve data. As files on IPFS are content-addressed (i. e. a file is addressed based on a hash of its content), the files' integrity is ensured by design, which reduces the trust requirements towards the nodes. Hence, IPFS is an ideal data storage system for decentralized apps hosted on a DL.

3 Related Work

To authorize issuers of paper-based education credentials, there are only very fragmented and country- or even sector-specific accreditation lists by authorities

on legitimate issuers, which moreover rarely include information for automated verification, such as public verification keys [23,33,14].

A number of projects [10,9,11] from public institutions and industry tackle the field of digital education credentials, with a strong focus on how to represent learning achievements. Some projects leave the assessment of the legitimacy of the credential's issuers to the verifier [41,29,24], while others propose centralized accreditation approaches [18,13,20,19], often limited to single countries or the EU and focusing merely on universities. In contrast, our approach is based on an open system, supporting the diverse needs of a global, growing and heterogeneous education world.

None of these projects are concerned with interoperability between different credential schemas. Although the W3C's VC standard [38] proposes a mechanism that would allow the transformation of a credential into another representation by its subject (using zero-knowledge proofs), we are not aware of any system that enables or supports the authenticated transformation of credentials between schemas directly by the verifier.

Research literature discusses further approaches for global identity and trust frameworks [27,28], with several publications focusing on DL technology: Alen et al. [4] introduce the concept of a decentralized public key infrastructure (DPKI). The concept proposes using a DL to map identifiers to key material, but does not define how to map entity identities to such identifiers. By suggesting central registrars, it effectively reuses the concept of a certificate authority. Trust over IP [12] combines DIDs, VCs and related technology and governance models into a stack compatible with our system. Alexopoulos et al. [3] discuss DL-based authentication and compare it to PKI and PGP's web of trust. Yakubov et al. [45] extend OpenPGP keys with the Ethereum address of their owner and provide a decentralized key server using smart contracts. Brunner et al. [8] provide a comparison of DL-based authentication frameworks, while Kuperberg [25] surveys blockchain-based identity management.

4 Architecture Overview

There are several *actors* in our system, also shown in Figure 1:

Student A natural person who earns credentials by studying and (eventually) graduating. The student is the data subject of a credential and thus needs a key pair and a corresponding DID registered on the ledger. Students may later act as applicants and present their credentials.

Credential Issuer Any institution that issues education credentials, e. g. a university, school, online learning provider or other training institution. A credential issuer has a key pair and a corresponding DID registered on the ledger.

Trust Certifiers All users who publish trust statements about the legitimacy (or illegitimacy) of issuers, the authenticity of transformation information, and their trust in other trust certifiers who publish trust statements. Credential

issuers, verifiers and any other stakeholders can also act as trust certifiers – enabling an open system to support a diverse education landscape.

Verifier An institution that accepts education credentials and wants to verify them, such as an employer or university. A verifier can also be a credential issuer (e. g. a university is usually both).

Creator The actor that creates and publishes the smart contract of our system, which maintains a web of trust on the DL. It does not matter who publishes the contract, as the contract's code is public and everyone is able to verify its behavior.

The actors deal with **Credentials** in W3C VC format. The education data inside the VC envelope is encoded following a schema chosen by the credential issuer. While the schema is stated in the credential, its interpretation is not necessarily known to the verifier.

Entities in our system provide **Trust Statements**, statements on the trustworthiness and legitimacy of others to help verifiers in their assessment process.

Definition 1 (Trust Statement). *A trust statement is a tuple of the form* $\langle C, \mathsf{LL}, \mathsf{LC}, \mathsf{c}, \mathsf{u}, \sigma, \mathsf{t}, I \rangle$*, where* C *is the trust certifier,* I *is the credential issuer being certified,* $\mathsf{LL} \in [-1, 1]$ *is the level of legitimacy* C *asserts for* I*,* $\mathsf{LC} \in [-1, 1]$ *is the level of confidence* C *has in the statements published by* I*,* c *is the context of the certification (credential or transformation),* u *is the identifier of what* I *is trusted to issue (type of credential or transformation) and* t *is a timestamp. Additionally,* σ *is a cryptographic assurance (digital signature) of the certificate's integrity and authenticity. Depending on the system chosen, levels are allowed to be any real value in the interval, one of several discrete values, or even just in* $\{-1, 0, 1\}$ *(with* 1 *denoting the highest trust level).*

The **Web of Trust** is a collection of trust statements.

Definition 2 (Web of Trust). *A web of trust is a directed, edge-labeled multigraph* $W = \langle V, E \rangle$*, where the vertices* V *are entities (e. g. educational institutions) and the edges* E *represent trust statements, which are consequently labeled as stated in the definition above. For edge labels in a web of trust, the trust certifier–credential issuer parameters have to match the edge's vertices, i. e. for each edge* e *from* v_1 *to* v_2*, the corresponding label of* e *has to be of the form* $\langle v_1, \mathsf{LL}, \mathsf{LC}, \mathsf{c}, \mathsf{u}, \sigma, \mathsf{t}, v_2 \rangle$*.*

To facilitate interoperability between credential schemas, trust certifiers also issue **Transformation Information**.

Definition 3 (Transformation Information). *Transformation information is a set of (machine-readable) transformation rules that define how to transform a credential issued in one credential schema into another schema. Transformation Information is encoded in an implementation-specific format.*

Analogous to the web of trust, a **Transformation Information Graph** is a collection of transformation information.

Definition 4 (Transformation Information Graph). *A transformation information graph is a directed, edge-labeled multigraph $TIG = \langle V, E \rangle$, where the vertices V are URIs of credential schemas and each edge $e \in E$ from vertex X to vertex Y represents a transformation for transforming a credential of schema X into a credential of schema Y. An edge also contains the DID of the entity that published the transformation information and is signed by it.*

The WoT and the transformation information graph are maintained by a **Registry**, a data structure on the DL. The registry is managed by a smart contract (SC) which provides a single point of contact for trust certifiers and verifiers who want to access or append to either graph by calling the *get* or *add* function on the contract, respectively. In addition, the registry utilizes IPFS to store the transformation information. While the transformation information itself is stored on IPFS, the respective edge in the transformation graph contains the IPFS address of the information and information about its trust certifier. An example setup is shown in Figure 3a.

A **Trust Policy** is defined by the verifier with rules on how to decide a credential issuer's legitimacy based on information obtained from the registry, i. e. trust statements and transformation information. Such a policy also defines suitable parameters for algorithms that find paths in the graphs and calculate legitimacy scores. Additionally, the trust policy contains the rules which are used to further check the credential's content, such as required study subjects or minimum grade point averages. The policy is defined locally by one or more domain experts. This approach allows verifiers to enforce their own rules and follow their locally relevant regulations.

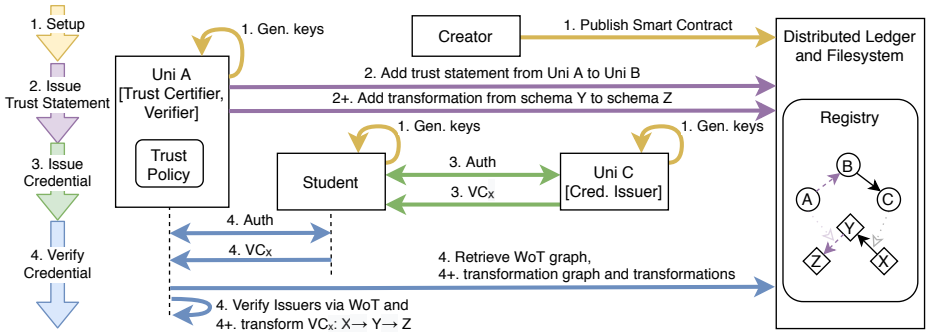


Fig. 1: Architecture and example dataflows of our system. University A, acting as trust certifier, adds an edge stating its trust about University B and publishes transformation information from schema Y to its schema Z. Later, University A, acting as verifier, verifies a credential issued by University C by additionally obtaining a previously registered edge from university B to C and a path from schema X to Y. They make sure that all transformation information is trusted by authenticating its issuer using the WoT.

5 Issuer Authorization

The first step needed to check a credential is to verify the legitimacy of the credential and its issuer using information published by other entities. This is also the basis for authenticated transformation of credentials between schemas, as described in the following Section 6.

Our approach can be split into four phases: (1) Initially, the registry and its smart contract are established on the DL. (2) The trust certifier identifies a credential issuer and publishes the certification information using the registry. (3) The credential issuer issues a credential to a student. (4) The student shows the credential to a verifier, who uses the information stored in the registry to verify the credential and the credential issuer's legitimacy. The main aspects of these phases are shown in Figure 1 and discussed in the following paragraphs, while a formal description is given in Figure 2.

(1) Setup: To set up the system, the creator publishes a smart contract on the DL which maintains the registry. This contract provides a single point to create or update (add) and retrieve (get) the lists of edges. Additionally, students, credential issuers and trust certifiers create their own key pairs and use them to establish their self-sovereign identities, i. e. derive and register DIDs. These DIDs with corresponding DID documents are registered on the DL to enable other parties to retrieve the respective public keys for signature verification processes.

(2) Issue Trust Statement: After having assessed a credential issuer's legitimacy (e. g. in a previously performed tedious manual process or using other channels), a trust certifier may share its decision with others by issuing a trust statement. Based on the previous assessment, the trust certifier chooses an appropriate level of legitimacy LL, which may also be a negative value if the trust certifier arrives at the conclusion that the credential issuer is an illegitimate institution. In addition, it adds the level of confidence in trust statements published by the credential issuer, represented by LC. The trust certifier publishes its assessment as a trust statement on the registry by setting the variables on the edge accordingly (cf. Section 4), using its private key to sign the edge, and publishes the signed edge by calling the smart contract through one of the DL's nodes. This creates a new edge in the registry's WoT graph pointing from the trust certifier to the prospective credential issuer. Optionally, the trust certifier may also issue transformation information or trust other entities to do so (cf. Section 6).

Revoke Trust Statement: As trust in other institutions changes and faulty entries occur, trust certifiers are able to add a new edge between the same vertices (but with a more recent timestamp) to the WoT, thereby altering the stated level of legitimacy or other attributes. Based on the timestamps, verifiers select edges at points in time based on the received credential and their trust policy.

(3) Issue Credential: After completing a course or graduating from a university, the student requests a VC for this accomplishment. Initially, the student proves their identity using their DID and private key. Then, the credential issuer loads the student's attributes from its student information system, encodes those attributes using a suitable JSON schema and places the encoded attributes into a VC. The credential issuer finally signs the VC and hands it to the requesting

(1) Setup: On any Creator: Generate $(\mathsf{pk_a}, \mathsf{sk_a}) \leftarrow \mathsf{KeyGen}()$, create smart contract for WoT registry, publish smart contract at DL and announce its address $\mathsf{Addr_{SC}}$ and code.
On all Students, Credential Issuers, and Trust Certifiers: Generate $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$, create DID from pk and register pk at DL.

(2) Issue Trust Statement: On Trust Certifier C: Verify identity and legitimacy of issuer and obtain issuer's $\mathsf{DID_{Iss}}$ (out-of-band). Create the WoT edge e, which includes $\mathsf{DID_C}$, $\mathsf{DID_{Iss}}$, the level of legitimacy as issuer LL, the level of confidence as trust certifier LC, the context c (set to *credential*), the credential's type u and the current time t. Sign $\sigma_\mathsf{e} \leftarrow \mathsf{SIG.Sign}(\mathsf{e}, \mathsf{sk_C})$, add the signature to the edge $\mathsf{e}' = \mathsf{e} \| \sigma_\mathsf{e}$ and publish the signed edge e' via $\mathsf{DL.Call}(\mathsf{Addr_{SC}}, \mathsf{add_{WoT}}, \mathsf{e}')$.
On all DL nodes: Receive the trust certifier's request and add e' to the WoT graph's list of edges.
Revoke Trust Statement: as Phase (2) with different values for LL and LC.

(3) Issue Credential: On Student S: Request verifiable credential at university and send $\mathsf{DID_S}$ as well as proof of ownership of $\mathsf{sk_S}$.
On Credential Issuer Iss: Verify student's identity and ownership of $\mathsf{DID_S}$. Encode the student's attributes A_S using $\mathsf{schema_{Iss}}$ into A'_S and generate credential cred, which includes the encoded attributes A'_S, student's $\mathsf{DID_S}$ and $\mathsf{DID_{Iss}}$. Sign the credential $\sigma_\mathsf{cred} \leftarrow \mathsf{SIG.Sign}(\mathsf{cred}, \mathsf{sk_{Iss}})$ and issue it to the student.

(4) Verify Credential: On Student S: Send $\mathsf{DID_S}$, proof of ownership of $\mathsf{sk_S}$ and cred' to the verifier.
On Verifier V: Load up-to-date $\mathsf{WoT} \leftarrow \mathsf{DL.Call}(\mathsf{Addr_{SC}}, \mathsf{get_{WoT}})$ and issuer's $\mathsf{pk_{Iss}} \leftarrow \mathsf{DL.Resolve}(\mathsf{DID_{Iss}})$, where $\mathsf{DID_{Iss}}$ from cred'. Verify that issuer was not revoked/expired, allowed to issue and that credential was really signed by that issuer by verifying $\mathsf{SIG.Verify}(\mathsf{cred}', \mathsf{pk_{Iss}}) = 1$. Find paths from verifier to issuer relevant w. r. t. cred' by $\mathsf{paths} \leftarrow \mathsf{pathfinder}(\mathsf{WoT}, \mathsf{DID_V}, \mathsf{DID_{Iss}}, \mathsf{c_{cred'}}, \mathsf{u_{cred'}})$. Authenticate edges in paths by verifying $\forall \mathsf{path} \in \mathsf{paths}, \forall \mathsf{edge\ e} \in \mathsf{path} : \mathsf{SIG.Verify}(\mathsf{e}, \mathsf{pk_C}) = 1$, where $\mathsf{DID_C}$ from $\mathsf{edge\ e}$ and $\mathsf{pk_C} \leftarrow \mathsf{DL.Resolve}(\mathsf{DID_C})$. Compute overall legitimacy of issuer and verify using policy that $\mathsf{calcscore}(\mathsf{paths}) \geq \mathsf{policy_{minscore}}$.

Fig. 2: Issuer authorization protocol.

student for their own use. The issuing of credentials is independent of the registry and a DL may only be needed to retrieve the student's public key via its DID for authentication.

(4) Verify Credential: When a student presents a credential to a verifier (e. g. within an application process), the verifier needs to verify it according to its trust policy. The verifier first retrieves the registry's WoT graph by calling the smart contract's $\mathsf{get_{WoT}}$ function using a DL node. On this graph, the verifier performs the path-finding algorithm defined in its policy. The verifier also verifies the signature of each trust statement in the trust paths with public keys obtained from the DL. If there are multiple paths towards a credential issuer, the verifier's trust policy may select which paths are relevant. This enables different verification

scenarios, such as "Was the credential issuer trusted at the point in time the credential was issued?" and "Is the credential issuer trusted right now?". On the identified paths, the verifier computes an overall "legitimacy score" and compares this score with the policy's requirements. Next, the verifier verifies the credential's signature with the public key that has been registered in the DL for the credential issuer's DID, including a revocation check by consulting the respective revocation registry. The verifier concludes the process by checking the credential with regard to a set of rules stated in the trust policy, such as the precise field of study or certain grade requirements. Optionally, if the verifier does not support the schema of the received credential, the verifier may engage in a transformation process and transform the credential into a supported schema as described in Section 6.

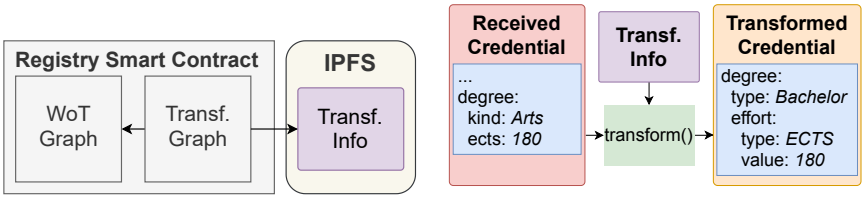
6 Credential Transformation

The second step during checking of a credential is to check the content of the credential using a local policy. The transformation steps described in this section help verifiers who are not familiar with the issued credential's schema and, therefore, need support to interpret the credential correctly.

We extend two of the previously described phases to enable transformations between schemas: (2+) Trust Certifiers knowing how to interpret a credential's schema and transform it into another schema publish this transformation information to the DL's registry to help others. For example, a university having dealt with credentials in a foreign schema previously (as verifier) may publish its transformation information into a local schema (as trust certifier). To establish trust in this transformation information, we build upon the web of trust approach as established in phase (2) above. (4+) After verifying the student's credential, the verifier transforms the credential into a supported schema by querying the transformation graph for suitable transformation information, verifying its authenticity and its publisher's legitimacy using the WoT graph.

Figure 3a illustrates how the WoT is used to authenticate the transformation information, and a formal view on the extended phases is given in Figure 4.

(2+) Issue Transformation: If a trust certifier has information about how to transform a credential from one schema to another, they can add this information



(a) Registry setup: Edges in the Transf. Graph are authenticated using the WoT.

(b) Transformation process using Transf. Info discovered using the Transf. Graph.

Fig. 3: Example transformation with information authenticated using the WoT.

to the transformation graph. As the transformation information may be too large, it is not stored on the ledger itself. Instead, the trust certifier publishes the transformation information on IPFS, which results in a content-dependent IPFS address (basically the information's SHA256 hash). This address is then added to the new edge and sent to the transformation graph by calling add_T.

Issue Trust Statement: Analogous to the assessment of a credential issuer's legitimacy, trust certifiers assess and share the legitimacy to publish transformation information and thereby state that they trust the other entity and its transformation information. This statement is published in the same way as in phase (2), with the context c set to *transformation*, and added to the WoT graph by calling $\mathsf{add}_{\mathsf{WoT}}$. The identifier u defines the respective schema.

(4+) Transform: After phase (4) has verified the credential as well as the credential issuer's legitimacy, the verifier retrieves transformation information from the registry by calling get_T and uses it to transform the credential into a schema known to the verifier's policy engine, as illustrated in Figure 3b. Initially, the verifier loads the transformation graph and finds a path from the credential's schema to the known schema. In each step, while traversing the path, the verifier verifies the signature on the edge and assesses the legitimacy of the edge's publisher using the WoT graph. It is also possible to define a rule in the policy requiring the creator of a transformation to possess a minimum legitimacy level and discard all other transformations. The verifier then loads the transformation from IPFS and verifies its integrity by computing the hash value and comparing it with the (content-dependent) address value stored on the edge. Afterwards, the verifier executes the transformation on the credential. If multiple relevant paths with transformation information exist, it depends on the trust policy which one is used – or if all are used and a trust decision is made by a human actor. Finally, the verifier obtains a credential represented in a supported schema, which simplifies further checks according to the trust policy (or manually by humans).

Of course, transformations between schemas have limitations. For example, if a target schema requires values that are not available in the source schema, a direct transformation might not be possible. In that case, the verifier or its policy have to decide whether the presented credential is considered sufficient or whether in light of these missing values (which are e. g. simply mapped to defaults), is is deemed insufficient for the assessment.

7 Prototype

This section describes the prototype implementation of our concept, which highlights the feasibility of our system.

Distributed Ledger (DL): We host our smart contract and its registry on an Ethereum-compliant DL. While the choice of permission model depends on the concrete use case, we used a public permissionless ledger for the proof-of-concept implementation, thus allowing participation without prior registration. While this is not a requirement, we host both graphs inside the same smart contract and thus on the same ledger. The registry was implemented as a smart contract

(2+) Issue Transformation: On Trust Certifier C: Define transformation information $\mathsf{transf}_{X\to Y}$ from schema_X to schema_Y and publish it on IPFS on address $\mathsf{Addr_T}$. Create transformation edge e_T, which includes $\mathsf{DID_C}$, the URIs identifying the source schema schema_X and target schema schema_Y, the IPFS address of the transformation $\mathsf{Addr_T}$ and the current time t. Sign the edge $\sigma_{\mathsf{e_T}} \leftarrow \mathsf{SIG.Sign}(\mathsf{e}_T, \mathsf{sk_C})$ and send it to contract by calling $\mathsf{DL.Call}(\mathsf{Addr_{SC}}, \mathsf{add_T}, \mathsf{e'_T})$.
On all DL nodes: Add $\mathsf{e'_T}$ to the transformation graph's list of edges.
Issue Trust Statement: as Phase (2) with the certification's context c set to *transformation* and the identifier u to a URI identifying a credential schema.

(4+) Transform: On Verifier V: Load up-to-date transformation graph $\mathsf{transg} \leftarrow \mathsf{DL.Call}(\mathsf{Addr_{SC}}, \mathsf{get_T})$ and find paths from issuer's to verifier's schema by $\mathsf{paths_T} \leftarrow \mathsf{pathfinder}(\mathsf{transg}, \mathsf{schema_{Iss}}, \mathsf{schema_V})$. For each path in $\mathsf{paths_T}$ and each edge in path, authenticate edge by using WoT, extract transformation address $\mathsf{Addr_T}$, load transformation $\mathsf{transf} \leftarrow \mathsf{IPFS.Load}(\mathsf{Addr_T})$ and transform credential $\mathsf{cred'} \leftarrow \mathsf{transform}(\mathsf{transf}, \mathsf{cred})$ for the next iteration. Each time, verify that cred fulfills $\mathsf{policy_{rules}}$ or reject stating a reason. Accept cred for further human processing.

Fig. 4: Credential transformation protocol.

using the Solidity smart contract language [16]. As an optional component, we operate a lightweight "listener" node inside the trust boundary of the verifier as an optional performance optimization to reduce the latency between the verifier's client and the DL node as well as to mitigate censorship attacks (cf. Section 8). This node receives and verifies all blocks of the ledger, but does not participate in the consensus protocol. Since the verifier is only concerned with the state of the registry's smart contract, all other transactions can be discarded, thus minimizing storage needs.

Distributed Storage Layer: As transformation information data is potentially too large to be stored directly on the DL, our implementation uses IPFS as the distributed storage layer due to its availability and maturity, although other distributed file systems could be used instead. Important properties are integrity protection of data and content-addressable file resolution.

Client Components: To add and retrieve edges from the registry, we implemented client components in JavaScript. All components use the Ethereum JSON RPC interface [15] to communicate with DL nodes. For the trust certifier, we use *web3.js* to create transactions and the *MetaMask* browser extension as the wallet to sign and send them to the DL. For the verifier, we implemented a client to retrieve and visualize the WoT graph and to compute paths in the graph, serving as the data source for the verification.

Transformation System: Transformation information is discovered using the registry's transformation graph and authenticated using the WoT graph, as explained in Section 6. The transformation information can be encoded by any means as long as the verifier is able to execute them. In our implementation, we used *jsonpath-object-transform* [26] and published corresponding templates en-

coded in JSON to IPFS. Other approaches to transform JSON between different schemas or to another representation (such as XML) include table-based transformations, more generic templating systems using JSONPath [22] and systems based on XML's XSLT (including its experimental variant JsonT [21]).

Trust Policy: The policy not only defines how to evaluate a credential's legitimacy, but also specifies a set of rules on whether the content of the credential is acceptable. For example, a university might want to restrict acceptable credentials to Bachelor's certificates during the application process for a Master's program. Our system does not hard-code such rules, but instead hands over all retrieved data to a policy system [6,30,2]. This policy system then executes a policy defined by domain experts, such as the university's registrar's office, providing them with a high-level way to define rules [31,42,43].

8 Discussion

This section discusses further aspects of our concept and prototype.

Local Trust: A tackled challenge is the authenticity and legitimacy of credentials that use unknown schemas. To avoid a signature becoming invalid due to transforming the credential, the verifier first verifies the signature of the original credential and only then converts the credential (based on trusted transformation information) locally. Both signature verification and credential transformation occur locally in the verifier's trust domain, and only trusted transformation information is used. Consequently, it is guaranteed that the transformed credential has the same meaning as the incoming signed one.

Revocation: Revocation of trust statements is an important feature in any trust management system [1]. In our proposed system, a trust certifier can revoke a trust statement at any time by issuing a new trust statement with a reduced legitimacy level – even with a negative value to explicitly declare mistrust in the credential issuer. Additionally, a credential issuer can revoke a credential after it has been issued, e. g. by publishing a corresponding statement to a revocation registry, which is checked by the verifier. Checks for both types of revocations need to be performed during credential verification by the verifier.

Smart Contract Security: Depending on the access model of the DL, many or even all entities can write data into the registry. Since all information published in the registry is signed by its publisher, this is not a problem for the authenticity of transformation information and trust statements. Nevertheless, an overfull registry could lead to performance issues during verification. To mitigate this issue, the smart contract serving the registry can be equipped with access control mechanisms. For example, it is possible to enforce that a credential issuer can only add edges which are outgoing from its own vertex. Preventing an attacker from adding invalid edges starting at the vertex of a legitimate entity keeps the path-finding algorithms from having to verify many invalid signatures to find the valid ones. Furthermore, it is possible to only allow adding edges to trust certifiers who are already part of the graph. While this keeps attackers away, it also hinders legitimate entities from joining the graph. Thus it depends on the concrete use case which mechanisms should be used.

Since a smart contract is code and code could contain bugs that malicious parties might be able to exploit [39], it is important to apply secure software development practices when adding access control mechanisms. This is even more important when considering that smart contracts cannot be changed after they have been deployed on a DL [36,37].

Operational Costs: To evaluate our concept, we deployed a smart contract that maintains a registry as described in Section 4, which manages the two graphs represented as a list of edges. The smart contract does not perform additional checks to verify the publisher, so a verifier needs to retrieve the full graphs and filter out any irrelevant or invalid edges. While a public DL enables an open system, all write operations on the ledger have a cost. In the Ethereum context, the cost of a contract call is measured in units called *gas* and depends on the required computational effort. Adding an edge to our registry costs about 78 000 gas, worth about US$ 10 in January 2021.[1] In contrast to write operations, read operations are free. As the majority of operations in our system are of the latter kind, the total costs are still relatively low.

Since costs of using a public ledger are hard to predict in advance, private or consortium ledgers represent alternatives. In such ledgers, members of a consortium operate all nodes of the ledger, removing the need for an incentive system like gas. This limits the expenses to the costs needed to host the nodes but restricts (write) access to consortium members.

Performance Evaluation: We evaluated graphs with up to 10 000 edges and measured the performance of different operations using an Ethereum-based DL. Adding a new edge to the graph is not a time-critical operation and only performed whenever a new trust statement is issued. The *add* call to our contract only took less than 1 second, while integration of the edge into the ledger depended on the ledger's block time (around 13 seconds for mainnet Ethereum [17]). Likewise, the retrieval of a graph with 10 000 edges from remote DL nodes (with *get*) took under one second. Performance is of even smaller concern if the verifier retrieves edges from a local "listener" node (cf. Section 7).

Sybil Attacks and Censorship: The existence of fake universities issuing (fake) credentials also makes the existence of fake trust certifiers issuing trust statements to (fake) universities plausible. However, since such fake trust certifiers are neither trusted by a verifier nor have a trust path from the verifier to them, these (fake) trust statements have no influence on a credential's verification. Although the decentralized and distributed nature of DLs provides resistance against censorship and denial-of-service attacks [3], a node might still provide bogus information to a verifier, hence a verifier needs to establish a trust relationship with at least one node. One way of doing this is having the verifier operate its own node as discussed in Section 7. Another way is to ask multiple or even all nodes to attest a certain set of edges represents the full graph and that no edges were censored.

[1] Computed by multiplying gas price (https://etherscan.io/chart/gasprice) and ether price (https://etherscan.io/chart/etherprice) on 22 January 2021.

9 Conclusion

In this paper, we introduced a system that simplifies the time-consuming and costly task of verifying credentials and the legitimacy of the credentials' issuer. While we placed the focus on the education domain, our approach can be generalized to other domains facing the same problems. Our open and decentralized system introduces a web of trust maintained by a smart contract on a DL. We build upon this trust assessment framework for transformation information, as well. Participants of our system publish transformation information on IPFS and a trust statement therefor on the DL. If confronted with a credential in an unknown schema, verifiers look up a transformation chain on the DL, as well as trust paths within the WoT to assess the legitimacy of these transformation steps. As a result, verifiers are able to transform received credentials to a supported data schema and automatically check their content. Finally, our implementation demonstrates the feasibility of our concept while the extensive discussion highlights additional benefits, such as the verifier's control over their trust decisions and the ledger's resistance against censorship.

References

1. Abraham, A., More, S., Rabensteiner, C., Hörandner, F.: Revocable and Offline-Verifiable Self-Sovereign Identities. TrustCom/BigDataSE 2020, IEEE (2020)
2. Alber, L., More, S., Mödersheim, S.A., Schlichtkrull, A.: Adapting the TPL Trust Policy Language for a Self-Sovereign Identity World. In: Open Identity Summit 2021. OID 2021, Gesellschaft für Informatik (2021), in press
3. Alexopoulos, N., Daubert, J., Mühlhäuser, M., Habib, S.M.: Beyond the Hype: On Using Blockchains in Trust Management for Authentication. pp. 546–553. TrustCom/BigDataSE/ICESS 2017, IEEE (2017)
4. Allen, C., Brock, A., Buterin, V., Callas, J., Dorje, D., Lundkvist, C., Kravchenko, P., Nelson, J., Reed, D., Sabadello, M., Slepak, G., Thorp, N., Wood, H.T.: Decentralized Public Key Infrastructure. White Paper, Rebooting the Web of Trust (2015)
5. Bear, J., Ezell, A.: Degree Mills: The Billion-Dollar Industry That Has Sold Over a Million Fake Diplomas. Prometheus Books (2012)
6. Becker, M.Y., Fournet, C., Gordon, A.D.: SecPAL: Design and Semantics of a Decentralized Authorization Language. J. Comput. Secur. **18**(4), 619–665 (2010)
7. Børresen, L.J., Meier, E., Skjerven, S.A.: Detecting Fake University Degrees in a Digital World. In: Corruption in Higher Education: Global Challenges and Responses, Global Perspectives on Higher Education, vol. 46, pp. 102–107. Brill | Sense (2020)
8. Brunner, C., Knirsch, F., Unterweger, A., Engel, D.: A Comparison of Blockchain-Based PKI Implementations. In: Proceedings of the 6th International Conference on Information Systems Security and Privacy. pp. 333–340. ICISSP 2020, SciTePress
9. Camilleri, A.F., Duffy, K.H., Otto, N.: Modeling Educational Verifiable Credentials. Draft community group report, W3C Verifiable Credentials for Education Task Force (2020), https://w3c-ccg.github.io/vc-ed-models, online, accessed on 22 January 2021
10. Camilleri, A.F., Tück, C.: Higher Education Interoperable Data Initiative (HEIDI). Living document (2020), https://heidirepo.github.io/HEIDI, online, accessed on 22 January 2021

11. Connecting Europe Facility: EBSI: Use Cases and Functional Documentation. https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITALEBSI/Use+Cases+and+Functional+Documentation (2020), online, accessed on 22 January 2021
12. Davie, M., Gisolfi, D., Hardman, D., Jordan, J., O'Donnell, D., Reed, D.: The Trust over IP Stack. IEEE Commun. Stand. Mag. **3**(4), 46–51 (2019)
13. Digital Credentials Consortium: Building the Digital Credential Infrastructure for the Future. https://digitalcredentials.mit.edu/wp-content/uploads/2020/02/white-paper-building-digital-credential-infrastructure-future.pdf (2020), online, accessed on 22 January 2021
14. ETER: European Tertiary Education Register. https://www.eter-project.com (2020), online, accessed on 22 January 2021
15. Ethereum: Ethereum JSON RPC API. https://eth.wiki/json-rpc/API (2020), online, accessed on 22 January 2021
16. Ethereum: Solidity Documentation. https://docs.soliditylang.org (2021), online, accessed on 22 January 2021
17. Etherscan: Ethereum Blocktime. https://etherscan.io/chart/blocktime (2021), online, accessed on 22 January 2021
18. European Commission: Europass Digital Credentials Infrastructure. https://ec.europa.eu/futurium/en/europass/europass-digital-credentials-infrastructure (2020), online, accessed on 22 January 2021
19. FutureTrust Consortium: Global Trust Service List. https://pilots.futuretrust.eu/gtsl (2020), online, accessed on 22 January 2021
20. Gräther, W., Kolvenbach, S., Ruland, R., Schütte, J., Torres, C., Wendland, F.: Blockchain for Education: Lifelong Learning Passport. In: Proceedings of the 1st ERCIM Blockchain Workshop. European Soc. for Socially Embedded Techn. (2018)
21. Gössner, S.: Transforming JSON. https://goessner.net/articles/jsont (2006), online, accessed on 22 January 2021
22. Gössner, S.: JSONPath - XPath for JSON. https://goessner.net/articles/JsonPath (2007), online, accessed on 22 January 2021
23. HEDD: UK Higher Education Degree Datacheck. https://hedd.ac.uk/about (2020), online, accessed on 22 January 2021
24. IMS Global Learning Consortium: Open Badges v2.0. Tech. rep. (2018), https://www.imsglobal.org/sites/default/files/Badges/OBv2p0Final/index.html
25. Kuperberg, M.: Blockchain-Based Identity Management: A Survey From the Enterprise and Ecosystem Perspective. IEEE Trans. Eng. Manag. **67**(4), 1008–1027 (2020)
26. Lane, D., Vontas, C., Rückstieß, T., Poggi, D.: jsonpath-object-transform. https://github.com/dvdln/jsonpath-object-transform (2017), online, accessed on 22 January 2021
27. Lee, A.J., Yu, T.: Towards Quantitative Analysis of Proofs of Authorization: Applications, Framework, and Techniques. In: Proceedings for the 23rd IEEE Computer Security Foundations Symposium. pp. 139–153. CSF 2010, IEEE (2010)
28. Li, N., Winsborough, W.H., Mitchell, J.C.: Distributed Credential Chain Discovery in Trust Management. J. Comput. Secur. **11**(1), 35–86 (2003)
29. MIT Media Lab Learning Initiative and Hyland Credentials: Blockcerts – An Open Infrastructure for Academic Credentials on the Blockchain (2016), https://www.blockcerts.org, online, accessed on 22 January 2021
30. Mödersheim, S.A., Schlichtkrull, A., Wagner, G., More, S., Alber, L.: TPL: A Trust Policy Language. In: Trust Management XIII – Proceedings of the 13th IFIP International Conference on Trust Management. pp. 209–223. Springer (2019)

31. Mödersheim, S.A., Ni, B.: GTPL: A Graphical Trust Policy Language. In: Open Identity Summit 2019. pp. 107–118. OID 2019, Gesellschaft für Informatik (2019)
32. Mühle, A., Grüner, A., Gayvoronskaya, T., Meinel, C.: A Survey on Essential Components of a Self-Sovereign Identity. Comp. Sci. Rev. **30**, 80–86 (2018)
33. Office for Students: OfS Register (Spreadsheet). https://apis.officeforstudents.org.uk/OfsRegisterDownload/api/Register/ (2021), online, accessed on 22 January 2021
34. Protocol Labs: IPFS Documentation. https://docs.ipfs.io (2021), online, accessed on 22 January 2021
35. Reed, D., Sporny, M., Longley, D., Allen, C., Grant, R., Sabadello, M.: Decentralized Identifiers (DIDs) v1.0. W3C working draft, W3C (2021), https://www.w3.org/TR/2021/WD-did-core-20210128/
36. Rodler, M., Li, W., Karame, G.O., Davi, L.: *Sereum*: Protecting Existing Smart Contracts Against Re-Entrancy Attacks. In: Proceedings of the 26th Annual Network and Distributed System Security Symposium. NDSS 2019, Internet Society (2019)
37. Rodler, M., Li, W., Karame, G.O., Davi, L.: EVMPatch: Timely and Automated Patching of Ethereum Smart Contracts. In: 30th USENIX Security Symposium. USENIX Security '21, USENIX Association (2021)
38. Sporny, M., Longley, D., Chadwick, D.: Verifiable Credentials Data Model 1.0. W3C recommendation, W3C (2019), https://www.w3.org/TR/2019/REC-vc-data-model-20191119/
39. Torres, C.F., Baden, M., Norvill, R., Pontiveros, B.B.F., Jonker, H., Mauw, S.: ÆGIS: Shielding Vulnerable Smart Contracts Against Attacks. In: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security. pp. 584–597. ASIA CCS '20, ACM (2020)
40. UK Department of Education: Higher Education Degree Datacheck. https://hedd.ac.uk/about (2020), online, accessed on 16 October 2020
41. W3C Verifiable Credentials for Education Task Force: vc-ed. https://w3c-ccg.github.io/vc-ed (2020), online, accessed on 22 January 2021
42. Weinhardt, S., Omolola, O.: Usability of Policy Authoring Tools: A Layered Approach. In: Proceedings of the 5th International Conference on Information Systems Security and Privacy. pp. 301–308. ICISSP 2019, SciTePress (2019)
43. Weinhardt, S., St. Pierre, D.: Lessons Learned – Conducting a User Experience Evaluation of a Trust Policy Authoring Tool. In: Open Identity Summit 2019. pp. 185–190. OID 2019, Gesellschaft für Informatik (2019)
44. Wright, A., Andrews, H., Hutton, B.: JSON Schema Specification. https://json-schema.org/specification.html (2020), online, accessed on 22 January 2021
45. Yakubov, A., Shbair, W., State, R.: BlockPGP: A Blockchain-Based Framework for PGP Key Servers. In: Proceedings of the 6th International Symposium on Computing and Networking Workshops. pp. 316–322. IEEE (2018)
46. Zwattendorfer, B., Zefferer, T., Stranacher, K.: An Overview of Cloud Identity Management-Models. In: Proceedings of the 10th International Conference on Web Information Systems and Technologies – Volume 2. pp. 82–92. WEBIST 2014, SciTePress (2014)