Towards Credential-based Device Registration in DApps for DePINs with ZKPs

Jonathan Heiss
Information Systems Engineering
TU Berlin, Germany
jh@ise.tu-berlin.de

Fernando Castillo
Information Systems Engineering
TU Berlin, Germany
fc@ise.tu-berlin.de

Xinxin Fan
IoTeX
Menlo Park, CA 94025, USA
xinxin@iotex.io

***Abstract*—Decentralized Physical Infrastructure Networks (DePINS) are secured and governed by blockchains but beyond crypto-economic incentives, they lack measures to establish trust in participating devices and their services. The verification of relevant device credentials during device registration helps to overcome this problem. However, on-chain verification in decentralized applications (dApp) discloses potentially confidential device attributes whereas off-chain verification introduces undesirable trust assumptions. In this paper, we propose a *credential-based device registration* (CDR) mechanism that verifies device credentials on the blockchain and leverages zero-knowledge proofs (ZKP) to protect confidential device attributes from being disclosed. We characterize CDR for DePINs, present a general system model, and technically evaluate CDR using zkSNARKs with Groth16 [1] and Marlin [2]. Our experiments give first insights into performance impacts and reveal a tradeoff between the applied proof systems.**

***Index Terms*—DePIN, Blockchain, Credential, Device, IoT, Registration, Verifiable, Zero-knowledge Proof, DApp**

I. Introduction

In untrusted environments like the Internet of Things (IoT), blockchains have manifested as a solution for managing devices and their data without introducing undesirable third-party dependencies or trust assumptions. Examples of such decentralized applications (dApps) include netting from smart meter data in energy grids [3], [4], product tracing through sensor measurements in supply chains [5], [6], and decentralized federated learning on healthcare data collected by wearables [7], [8].

More recently, Decentralized Physical Infrastructure Networks (DePINs) [9] have emerged as a class of dApps that add crypto-economic mechanisms to the dApp's smart contracts to incentivize the provisioning and consumption of device-enabled services. This has led to the formation of large decentralized networks of devices that collectively offer project-specific services, providing viable alternatives to centralized service models. Examples of industrial projects utilizing DePINs include IoTeX for sensing services [10], Helium for connectivity services [11], StreamR for data streaming services [12], and Acurast for computational services [13].

In DePINs, token-based incentive schemes represent the key element to governing and establishing trust in the DePIN's service model. Such schemes rely heavily on off-chain data from the devices to trigger token issuance on completed service provisioning. However, the reliance on off-chain data represents a security threat. Since blockchain security guarantees do not extend beyond the smart contract's application logic, malicious actors can corrupt data to trigger unwarranted token issuance. This makes *trustworthy data and service provisioning* indispensable for the success of DePINs.

Trustworthy data provisioning is challenging as devices typically do not communicate directly with smart contracts but data provisioning is intermediated by device owners or third-party oracles [14]. To prevent data corruption authenticity proofs created by the devices are verified on the blockchain using the devices' public keys. Additionally, zero-knowledge proofs (ZKP) can be employed for trustworthy preprocessing [15]–[18] enabling end-to-end verification of *data in use* between devices and smart contracts.

The *trustworthiness of the service provisioning* is hard to guarantee without measures to verify or enforce service qualities. In DePINs, such qualities strongly depend on the capabilities and attributes of the devices providing specific service types. Service quality assurance, consequently, requires devices to meet certain attribute-based criteria. Computation services may require a minimum of CPU cycles or memory capacity, connectivity services may require a certain level of bandwidth or range, and sensing services may require specific sensor capabilities or devices to be placed in certain locations.

Transparently enforcing such conditions on device attributes in decentralized settings is challenging and requires an appropriate model for managing device identity attributes. For that, concepts of the Self-Sovereign Identity (SSI) paradigm represent a promising approach promoting decentralized and user-centric management of identity-related data. Applied to DePINs, device manufacturers and other identity providers can issue verifiable credentials (VCs) [19] that attest to device attributes, which devices or their owners can present to third-party consumers.

A problem for decentralized attribute verification is that device credentials often contain confidential information that must not be made public, like private residence locations or security-relevant parameters like software versions. Blockchain-based verification of signature-based attestations guarantees transparency but publicly exposes these attributes violating confidentiality requirements. Alternatively, off-chain verification which is the common practice in many DePIN projects introduces undesirable trust assumptions and risks

excluding eligible devices or registering non-eligible ones, thereby undermining promised decentralization.

Addressing this conflict of confidentiality and transparency, we propose a mechanism for *credential-based device registration* (CDR) for dApps in DePINs that enables non-disclosing verification of device credentials on the blockchain using ZKPs. Verified devices are registered on-chain with their public key which is later used to validate the authenticity of data received from the devices. The mechanism builds upon and extends the W3C verifiable credential model [19] that best fits the contextual demands of dApps in DePINs. In this preliminary work, we make three individual contributions:

- We present a system model for DePINs that supports integration with the W3C VC model and the proposed CDR mechanism. The model is underpinned with examples of registration conditions, device attributes, and VC issuers facilitating its instantiation.
- We present a device registration mechanism that leverages verifiable device credentials to transparently decide registration conditions on the blockchain. To hide confidential device attributes, ZKPs are used for off-chain pre-processing yielding non-disclosing and on-chain verifiable proofs of registration conditions.
- We evaluate the system through a prototypical implementation using ZoKrates [20] for zkSNARK creation and verification on Ethereum [21]. We conduct initial experiments on test credentials with Groth16 [1] and Marlin [2]. The results give first insights into the method's performance behavior.

II. Preliminaries

As central concepts, this work relies on *verifiable credentials* (VC) and *zero-knowledge proofs* (ZKP).

A. Verifiable Credentials

The W3C recommendation for VC [19] advocates for a user-centric identity management framework. As depicted in Figure 1, identity attribute claims are issued by a trusted *issuer* as VCs to the *holder*, who then securely stores them, e.g., in a digital wallet. A *credential* (CR) contains claims of an issuer about the device that consists of a 3-tuple comprising subject, attribute, and value, e.g., $(\text{Alice}, \text{lives\_in}, \text{NYC})$. A VC adds authorship of issuers through attestations to a credential's claims that can be cryptographically verified. The holder can then independently present a selection of these verifiable attribute claims as a *Verifiable Presentation* (VP) to a *verifier*. VPs can be created with ZKPs to hide confidential information from the verifier. Verifiers can specify the VP through a *Verifiable Presentation Request* (VPR) [22] according to the needs of the application at hand. Unlike the focus of this paper, the VC model assumes the verifier operates off the blockchain. Blockchains are utilized solely to implement *Verifiable Data Registries* (VDR), which store public artifacts such as identifiers, public keys, or *credential schemas* (CS) that define the structure and verification process of VCs.

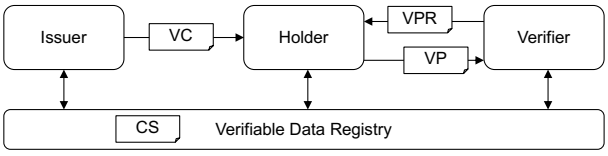

Fig. 1: Verifiable Credential Model based on [19]

B. Non-Interactive Zero-knowledge Proofs

Non-Interactive Zero-knowledge Proofs (NIZK) allow a prover to convince a verifier of a statement in one message (non-interactive) and without disclosing any confidential information despite the statement itself. Zero-knowledge succinct non-interactive arguments of knowledge (zkSNARKs) represent a specific category of NIZK known for their compact proof sizes and efficient verification times. The zkSNARKs procedure can be conceptualized in three main operations:

- The *setup* ($setup(ecs, srs) \rightarrow (PK, VK)$) generates a public asymmetric key pair derived from the executable constraint systems (ecs) and structured reference string (srs). Both proving and verification keys (PK, VK) are tied to the ecs which encodes the program logic in a provable representation. This process assumes a secure disposal of the srs to prevent the creation of fake proofs.
- The *proving* ($P(ecs, x, x', w, PK) \rightarrow \pi$) occurs in two steps: Firstly, a witness w is created by executing the ecs on the public inputs x and the private inputs x' constituting the proof arguments. The witness w signifies a valid variable assignment for the ecs inputs. Subsequently, the proof π is generated from the witness using the PK.
- The *verification* ($V(\pi, x, VK) \rightarrow \{0, 1\}$) evaluates the proof π and the public inputs x using the verification key VK.

ZkSNARKs facilitates Verifiable Off-Chain Computing [23] (VOC) which helps to overcome the privacy and scalability limitations of blockchains by offloading computation without compromising the blockchain's integrity. VOC is technically supported by *ZoKrates* [20], a language and toolbox that facilitates the development of zkSNARKs-based VOC for the Ethereum blockchain [21].

III. Model

To set the scenes, in this section, we first characterize credential-based device registration (CDR) through examples of device attributes, registration conditions, and credential issuers, then we present a general system model for CDR in dApps, and finally outline the threat model and associated objectives for designing a CDR mechanism in DePINs.

A. Characterizing Credential-based Device Registration

We characterize CDRs by giving examples of *conditional checks* executed on *device attributes* attested to by *credential issuers*.

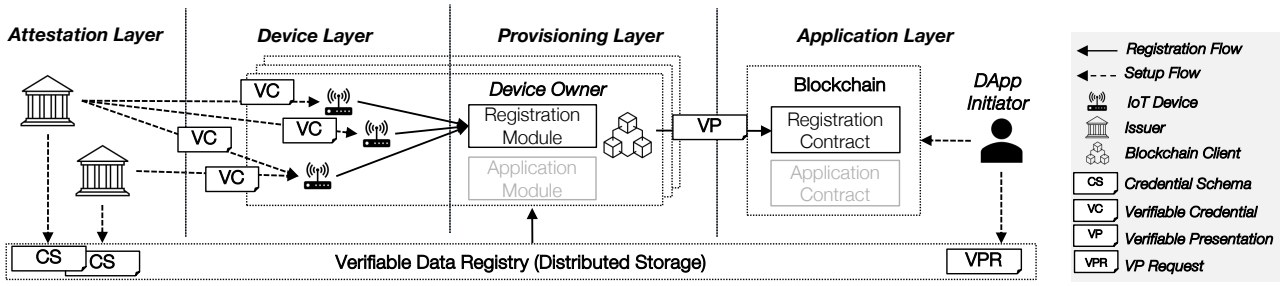

Fig. 2: System Model for DApps in DePINs Enabling Device Registration with Verifiable Credentials

1) Conditional Checks: Registration conditions can be realized with equality, range, membership, and time-dependent proofs [24].

- *Equality checks* are used in dApps if a device attribute must be equal to a predefined value.
- *Range checks* are used in DApps if a numeric attribute is required to be in a specific range indicated through an upper and/or lower bound.
- *Membership checks* are used in dApps if it is required that a device's attribute value val is in a predefined finite set $S = s1, s2, ..., sn$ such that $val \in S$.
- *Relative time-dependent checks* are used in dApps if a date- or time-based attribute is required to be in a range that has boundaries relative to the current date or a timestamp.

2) Device Attributes: Such registration conditions rely on device attributes of different types. The following examples, build upon and extend the collection presented in [25].

- *Identity attributes* can be categorized in *static attributes*, e.g., public key, serial number, manufactured date; and *dynamic attributes*, e.g., firmware version, last_updated_on, owner_id.
- *Capability attributes* describe available resources and functionality of the devices regarding *communication*, e.g., wired, wireless, or satellite; *computation*, e.g., clock speed, number of cores; *memory*, e.g., size.
- *Configuration attributes* can comprise the device *thresholds*, e.g., max and min values; *security parameters*, e.g., elliptic curve types or hash algorithm; *communication types*, e.g., scheduling of cron job intervals.
- *Installation attributes* further describe the installation or onboarding process. This may concern the *installer*, e.g., installer_id, certificate, association; and the *installation*, e.g., time, location, sealing.

3) VC Issuers: Device attributes can be attested to by different types of issuers as presented in [25].

- *Manufacturers* have access to critical device information and control large parts of the device's lifecycle which allows for attesting to many attributes.
- *Regulators* can establish and enforce industry standards and regulations, e.g., for medical devices or smart meters.
- *Service providers* can attest to attributes associated with their offered service, e.g., device installation or firmware updates.
- *Device owners* can attest to a variety of attributes including self-attesting the device ownership.

B. System Model

While previous examples are intended to help design CDR conditions, their integration with the W3C VC model and DePIN applications can be challenging. For that, we propose a general system model for device registration in dApps that integrates the W3C VC model presented in Section II-A. As depicted in Figure 2, the system consists of four layers.

1) Attestation Layer: On the attestation layer, issuers attest to the attributes of the device. Device attributes can be attested to by different issuers. For example, the same industry certificate may be issued by different accredited service providers or authorities. The resulting verifiable credentials are provisioned to the devices where they are protected against unauthorized access. The issuer uses the *verifiable data registry* (VDR) to publish the corresponding *credential schema* (CS).

2) Device Layer: On the device layer, devices collect data from the service provisioning relevant to the smart contract-based application logic, e.g., token issuance in DePINs. Devices are characterized by attributes that are subject to the CDR condition. Such attributes are attested to by issuers and contained in a *verifiable credential* (VCs) which can only be accessed by the device owner and the associated issuers. To include resource constraint devices, we assume separate the registration and application logic as well as the interaction with the smart contracts to the device owners who are assumed to act on behalf of the devices.

3) Provisioning Layer: On the provisioning layer, device owners provision data obtained from the devices to the smart contracts through a *blockchain client* using their blockchain account address. The *application module* (AM) is responsible for tasks related to data provisioning once a device is registered which may include a pre-processing of sensor data [18] or the creation of a service provisioning proof. The *registration module* (RM) is used for device registration and the focus of this work. Here, the *verifiable presentation* (VP) is created from the devices' VC according to the *verifiable presentation request* (VPR).

4) Application Layer: The application layer consists of two types of smart contracts running on a blockchain infrastructure. They are deployed by the *dApp initiator* who represents

the project provider in DePIN projects. *Application contracts* implement context-specific application logic that relies on the off-chain device data, e.g., the token issuance. *Registration contracts* check the CDR condition that devices must satisfy to be accepted as data sources for the application logic. The registration contract takes the VP as input which is created by the device owner according to the VPR specified by the initiator.

C. Threat Model and Challenges

In the previous model, we expect attacks from device owners and the dApp initiator whereas issuers are assumed to be trusted, following the W3C VC model [19]. Device owners may want to corrupt the registration mechanism by submitting VPs containing false claims to obtain tokens from services provided by non-eligible devices. Where possible, the dApp initiator may try to register non-eligible devices acting as device owners. Both, device owners and the dApp initiator may also try to obtain access to confidential device attributes. To prevent such threats, we derive the following objectives for a CDR mechanism in the described system model:

- *Verifiability*: The device registration must be verifiable by all device owners involved in and affected by the dApp. This includes the correctness of the issuers' attestations and the validation of the registration condition.
- *Confidentiality*: Device attributes must not be disclosed to anyone but the associated device owner. This means that neither the attestations nor the registration condition can be verified directly by other device owners or be verified on the blockchain.

IV. System Design

To achieve previously formulated objectives, in this section, we propose a mechanism for credential-based device registration (CDR) that leverages zkSNARKs to make the off-chain validation of CDR conditions verifiable on the blockchain without disclosing confidential device attributes. Distinctive artifacts are the *zero-knowledge verifiable presentation request* (zkVPR) created by the initiator that helps the device owner to create a *zero-knowledge verifiable presentation* (zkVP) which is validated by the registration contract as a non-disclosing registration request. On successful verification of the zkVP, the device's public key ($pubk_d$) is registered on-chain and can later be used to authenticate device-generated data submitted to the dApp. The procedure is depicted in Figure 3 and consists of three phases.

A. The *attestation* is considered a pre-requisite executed by the issuers who create the VCs by signing the device attributes.
B. The *setup* is executed once per application by the initiator who creates and deploys the zkVPR and the registration contract.
C. The *registration* is executed for each device. It consists of a proving where the device owner creates the zkVP and the verification where the zkVP is verified by the registration contract.

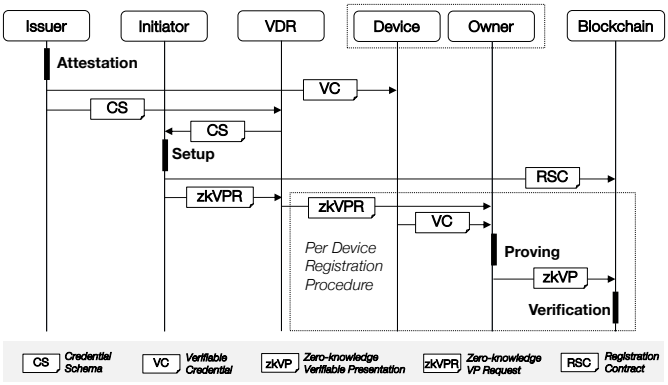

Fig. 3: Device Registration Procedure

In the following, we describe each phase in more detail.

A. Attestation

During the attestation, the issuer creates a verifiable credential (VC) by signing each claim of a credential (CR) individually with the *issuer secret key* $seck_i$: $Attest(CR, seck_i) \rightarrow (VC)$. This results in a VC consisting of a set of *verifiable claims* (vcl) ($VC = \{vcl_1, ..., vcl_n\}$) each represented as a claim-signature pair ($vcl = \{cl, sig\}$).

We assume that each device has a device-specific public key ($pubk_d$) that could be attested to by a suitable issuer, e.g., a public key authority. The $pubk_d$ will be used on-chain to verify authenticity proofs of the device created with the corresponding device secret key ($seck_d$). We treat the certificate as a VCL that contains $pubk_d$ as the attribute.

The resulting VC is securely stored on the device and the corresponding CS is published on the *Verifiable Data Registry* (VDR), a distributed storage system.

B. Setup

For the setup, the initiator creates and deploys the *zkVPR* and the *registration contract*. A zkVPR consists of the *zero-knowledge proof specification* ($zkSpec$), the *proving key* (pk), and a reference to the corresponding CS. Additional meta information can be added. To deploy the registration contract, the *verification key* (vk) is required to validate the correctness of the zkVP. The setup can be described in three steps.

1) Proof Specification: The zkSpec should allow the device owners to create the zkVP and, for that, it specifies the logic of the CDR condition to be proven in zero-knowledge and the required inputs. CDR conditions typically consist of two operations, both executed for each relevant VCL:

- *Authenticity Check*: The issuer's signature (sig) is verified using the issuer's public key (pk_i): $Verify(sig, cl, pk_i) \rightarrow \{1, 0\}$
- *Conditional Check*: As described in Section III-A1, an attribute-specific registration condition is checked using some auxiliary data (aux) like thresholds for range proofs or candidate lists for membership proofs: $Check(att, aux) \rightarrow \{1, 0\}$

Public inputs (x) are pk_i and aux whereas private inputs (x') are sig, cl, and att. While the composition of VCLs can vary per application, each zkSpec of a CDR condition must contain the device's public key ($pubk_d$) as a public input to bind the zkVP and its on-chain validation to the device. The $pubk_d$ is later added to the device registry of the registration contract. If both checks pass for all VCLs, the device's public key is returned to bind the device attributes to the key.

2) Key Generation: The initiator creates the zero-knowledge *proving key* (pk) and the *verification key* (vk). For that, first, the zkSpec is compiled into an executable constraint system (ecs) to enable the assertion of computational correctness through a zkSNARK. Using the ecs, the device owner generates pk and vk as described in Section II-B: $KeyGen(ecs, srs) \rightarrow (pk, vk)$. The keys are bound to the ecs and enable the device owner to create a verifiable $zkSpec$-specific zkVP with the pk that can be verified by the registration contract with the corresponding vk.

3) Deployment: The initiator publishes the $zkSpec$, the pk, and the CS-reference as zkVPR to the VDR where it becomes accessible by the device owners. The zkVPR-reference and the vk are integrated into the registration contract which is finally deployed to the blockchain.

C. Registration

The registration consists of the off-chain proving by the device owner and the on-chain verification by the registration contract.

1) Proving: The device owner first collects the necessary information and artifacts, that is, the zkVPR-reference from the registration contract and the zkVPR and CS from the VDR (the CS-reference is contained in the zkVPR). The *zkSpec* and the CS provide the necessary information to create the required zkVP. Accordingly, the device owner accesses the requested verifiable claims (vcl) from the device's secure storage which represents the private inputs (x') to the proving.

To create the zkVP, the device owner re-compiles the zkSpec into a ecs and creates the zkVP in two steps as described in Section II-B. First, the witness is generated by executing the ecs using the vcl as private inputs x' and the auxiliary data aux and public keys ($pubk_d, pubk_i$) as public inputs x. Second, the zkVP is created on the witness w using the pk contained in the zkVPR. A successful execution results in the zkVP consisting of $pubk_d$, aux, and the proof of computational correctness π.

2) Verification: The verification is executed by the registration contract on the zkVP obtained from the device owners as a registration request. The registration contract implements (1) the device registry, and functionality for (2) the ZKP verification, and (3) additional checks in the public inputs (x). It is executed in two steps:

- *Proof Verification*: The zkVP's correctness is validated using the verification key, public inputs, and proof: $Verify(\pi, vk, pubk_i, pubk_d, aux) \rightarrow \{1, 0\}$.
- *Input Verification*: The applied public inputs ($pubk_i, pubk_d, aux$) are checked against a predefined list of inputs stored in the registration contract.

If both checks pass, the device's public key is added to the device registry. With that, it can be enforced that only registered devices can call functions in the application contract. As a provisioning condition, devices must create a signature over the device-generated data using the secret key that matches a public key in the registry.

V. Evaluation

Given a detailed specification of the credential-based registration (CDR) mechanism, in this section, we describe our prototypical implementation and present our initial experiments.

A. Implementation

To demonstrate the technical feasibility of our proposal, we prototypically implement the CDR mechanism. The *attestation* is realized with a Python script that creates EdDSA signatures over the test credentials used for the experimentation. It should be noted that the elliptic curve applied for signature creation must be supported by the proof system and the verification environment. Respecting these dependencies, we use the babyjubjub curve (ALT_BN128) that is supported by the Ethereum Virtual Machine [21] and, hence, allows for verification of babyjubjub-based SNARKs. The *setup* and *proving* are realized with ZoKrates [20]. We implement the proof specification in the ZoKrates DSL resulting in a human-readable and small-sized artifact that is suited for efficient sharing and allows the device owner to understand and double-check the proof logic. Furthermore, we use the ZoKrates Command Line Interface for the compilation into a ecs, key generation, witness computation, and proof generation. For *verification*, we use the Solidity verifier smart contract generated by ZoKrates. It implements the routines required to verify the ZKP in the verytx() function using the integrated verification key. Smart contracts are hosted on a locally simulated Ethereum blockchain using Hardhat[1] test suit.

B. Experimentation

To obtain insights into the practicality of the system, we conduct initial experiments on our prototypical implementation.

1) Objective: The usage of zkSNARKs and blockchains adds considerable performance overhead to the (zk)VPR creation, the (zk)VP creation, and the (zk)VP verification. Such overheads may lead to unacceptable resource requirements, execution times, or transaction costs preventing adoption in practical settings. The objective of these experiments is to get initial insights into how the usage of zkSNARKs for anonymous device credentials in dApps negatively impacts such performance-related qualities.

[1]https://hardhat.org/hardhat-runner/docs/getting-started

TABLE I: Experimental Results

Proof	Scheme	TX Cost (Gas)	Witness (s)	Setup (s)	Proof (s)	Compiled (MB)	PK (MB)	VK (KB)
Range	Groth16	595 k	3	3	3	400	59	8
Membership	Groth16	623 k	5	7	6	752	126	8
Equality	Groth16	495 k	2	3	3	400	59	8
Range	Marlin	1007 k	2	1322	39	453	2322	9
Membership	Marlin	1035 k	4	2674	79	906	4624	9
Equality	Marlin	889 k	2	1338	38	453	2322	9

2) Design: To achieve this objective, we implement several zero-knowledge proof specifications (zkSpec) using different registration conditions, compile them into zkVPRs, execute them on typical device attributes, and verify the resulting zkVPs in an Ethereum Virtual Machines [21]. zkSpecs define the (1) validation of the issuer's signature and (2) an attribute-based condition for each device attribute. In the experiments, we use three typical conditions as described in Section III-A1:

- First, we use a *range check* to validate that only devices are registered that have firmware with a minimum version number. The private device attribute is the firmware number and the public threshold is the minimum version.
- Second, we use a *membership check* to validate that only devices within a certain regional range are registered. The private device attribute is a postcode representing the device's location and the range is defined by a public list of permissible postcodes.
- Third, we use an *equality check* to validate that only devices with specific measurement types are registered. The device attribute is a code representing the device's measurement type. It is checked against a predefined type.

In anticipation of a tradeoff between performance and security, we use two different proof systems to technically realize zkSNARKs: First, we use *Groth16* [1] which is well-established and known to be efficient but relies on a trusted setup as described in Section II-B. Second, we use *Marlin* [2] which mitigates trust assumptions from the setup through a universal and updatable structured reference string (srs) which, however, is expected to cause a performance loss.

We execute each experiment using test credentials on a MacBook Pro (Model Identifier: Mac14,10, Model Number: MNW83CI/A) with an Apple M2 Pro chip (12 cores: 8 performance and 4 efficiency), 16 GB of memory, and System Firmware Version 10151.101.3. For each proof and scheme combination, we measure the transaction cost in terms of Gas, the time taken to generate the witness, setup time, and proof generation time in seconds. Additionally, we report the sizes of the compiled ecs and proving key (PK) in megabytes (MB), and verification key (VK) in kilobytes (KB).

3) Results: As depicted in Table I, the experimental results show the trade-offs between the proof schemes. Groth16 exhibits lower indicators overall when comparing the same proof with a different scheme, with the exception of the witness computation. On average, comparing Groth16 with Marlin, there is an increase of 71% for Gas, 40930% for setup and 1200% for proof times, 16.75% for compiled ecs size, 3698% for private key size, 12.5% for verification key size and a decrease of 20% for witness computation time.

To assess the practical implications for DePINs, we can distinguish between *one-time and recurring operations*. The setup is executed once per proof specification resulting in one zkVPR. A single zkVPR may be sufficient for a DePIN project if it contains all necessary registration conditions. However, when the device registration conditions change, a new zkVPR must be created resulting in another setup execution. While the setup using Marlin leads to long execution times, such overheads can be considered insignificant in practice as zkVPRs are not expected to change frequently.

Recurring operations, i.e., witness computation, proof generation, and verification, are executed once per device registration. The combined off-chain operations of witness computation and proof generation do not exceed 11 seconds for Groth16 but with Marlin they lead up to 83 seconds for membership proofs. Similarly, transaction costs increase when using Marlin. While both, off-chain execution times and on-chain transaction costs, are bearable for single devices registration even for Marlin, they may represent a hurdle for time-critical registration of large numbers of devices.

VI. Discussion

In this Section, we revisit security objectives from Section III and discuss yet unaddressed issues of the system.

A. Trusted Setup

By using Marlin [2] as a proof system, we can reduce trust assumptions from the setup and protect against attacks of the dApp initiator. With Marlin, the srs is updated over time by multiple parties with the security assumption that at least one participant in each update phase is honest. Each update enhances the trustworthiness of the srs because it adds layers of contributions from various parties. In the proposed system, device owners can participate in the updating procedure to be certain about the trustworthiness. However, as our experiments show, these improved security guarantees come at the cost of higher transaction costs and execution times.

Beyond that, such attacks can be prevented through an adjustment of the setup phase, e.g., employing a blockchain-based setup ceremony as proposed in [15] or leveraging secure multi-party computation (sMPC) as proposed in [26]. Furthermore, different proof protocols could be applied that do not require a trusted setup like STARKs [27] which, however, are more expensive to verify. As another approach, some zero-knowledge Virtual Machines (zkVM) like Risc Zero combine

STARKs and SNARKS to prove the correct execution of the VM's instruction. In contrast to a per-application setup for SNARKs, zkVMs rely on a single setup for the zkVM that is often well documented[2].

B. Replay Attacks

Replay attacks pose another significant problem for the system. After on-chain verification, proofs become publicly accessible, allowing anyone on the blockchain to resubmit already accepted zkVPs to impersonate another identity and gain unauthorized registration. To mitigate this issue, uniqueness proofs can be used as proposed in [24] which enable the dApp to detect repeated submissions of the same proof by different users.

C. Revocation

The issuer may need to revoke a credential or a credential may be subject to change, e.g., if a device's firmware is updated, the version number changes. Although revocation is not covered in this work, we suggest using specialized blockchain-based revocation systems as an extension to our credential on-chaining system, similar to the methods proposed for educational credentials in [28]. Additionally, in some instances, revocation can be substituted with expiration dates on credentials, which can be implemented using relative time-dependent proofs as proposed in [24].

D. Public Key Disclosure

The device registry of the registration contract contains the public keys of all registered devices ($pubk_d$). The disclosure of the $pubk_d$ on the blockchain opens the opportunity for attacks on the device, e.g., key search attacks. To protect the $pubk_d$, we propose the following extension: Only a hash-based commitment to $pubk_d$ is stored in the registry. To authenticate, the device owner creates a zero-knowledge proof that encodes (1) the verification of the signature using $pubk_d$ as private input and (2) proves that the $pubk_d$ belongs to the commitment which is treated as public inputs. On-chain, the correctness can be verified through a successful proof verification and a comparison of the public input with the corresponding $pubk_d$ commitment of the registry.

VII. Related Work

To our best knowledge, we provide the first credential-based device registration procedure for dApps and DePINs. However, our research intersects with various studies on the applicability of self-sovereign identity (SSI) concepts in the Internet of Things (IoT), the use of verifiable credentials (VC) in blockchain-based decentralized applications (dApps), and zero-knowledge proof (ZKP)-based anonymous credentials.

SSI for IoT: The application of self-sovereign identity for IoT devices has been examined in several studies. Fedrecheski et al. [29] compare existing models for device identities, attributes, and key management, including PGP, X.509, and SSI. Dayaratne et al. [25] elaborate on the potential of SSI for IoT through a taxonomy that helps compare different device issuers, IoT use cases, and device lifecycle management. Gebresilassie et al. propose and evaluate an SSI-based system for IoT in the context of a car rental use case [30]. Additional use cases for SSI in industrial IoT are discussed by another study [31]. In these approaches, blockchain technology is used as a Verifiable Data Registry (VPR) rather than as an application platform. In contrast, our work focuses on smart contract-based validation of device credentials, enabling their integration in dApps.

SSI in dApps: Several studies address the use of device credentials in dApps. The DIAM-IoT framework, introduced by Fan et al. [32], is a decentralized Identity and Access Management (IAM) system designed for data sharing in the IoT. Luecking et al. [33] propose an SSI-based system for IoT devices that uses blockchain technology to host a reputation system for these devices. Another study [34] utilizes decentralized identifiers and VCs in blockchain-based data trading systems to authenticate users and prove data ownership. While these works extend the use of blockchains beyond VDR functionality, they do not provide smart contract-based validation of device credentials.

Anonymous Credentials in dApps: ZKP-enabled anonymous credentials are explored for off-chain verifiers in various industrial projects like Privado (former PolygonID) [35] or Hyperledger AnonCreds [36]. Research on smart contract-enabled on-chain verifiers is conducted only in a few proposals. Yin et al. [37] propose an identity system for IoT based on consortium blockchains that use commitments and ZKPs to protect confidential attributes on-chain. While this system supports on-chain credential verification, it is not suitable for integrating verification results in dApps without trusted intermediaries. Muth et al. [38] demonstrate how CL-signature-based anonymous credentials from the Hyperledger Indy ecosystem can be verified by smart contracts running in the Ethereum Virtual Machine. Heiss et al. [24] continue this work by applying zkSNARK-based verifiable off-chain computation to enable privacy-preserving credential verification in dApps. These approaches are closely related as they verify credentials on the blockchain, but they do not focus on device credentials, which is the primary focus of our research.

VIII. Conclusion

In this paper, we introduced a credential-based device registration (CDR) mechanism for dApps in DePINs. Our mechanism uses zkSNARKs to allow device owners to create zero-knowledge verifiable presentations (zkVPs) based on requests (zkVPRs) from dApp initiators. These zkVPs can be verified on the blockchain without revealing confidential device attributes, ensuring transparent verification by other device owners. We provided a detailed characterization of CDRs, including registration conditions, device attributes, and credential issuers, and presented a system model integrating CDRs with the W3C VC model. We implemented the CDR using ZoKrates [20] to create and verify zkSNARKs-based zkVPRs and zkVPs on the Ethereum blockchain. Our evaluation with

[2]https://dev.risczero.com/api/trusted-setup-ceremony

Groth16 [1] and Marlin [2] on different registration conditions gave first insights into the performance impact of applied proof systems and highlighted a tradeoff between security and efficiency. While we see CDRs as a crucial mechanism to establish trust in DePINs, we consider this work preliminary and plan to continue our research. In addition to the discussed security concerns, future work will address the integrability with existing DePIN technologies like W3bstream, the applicability of alternative proving environments like zero-knowledge virtual machines, architectural variants considering self-sovereign devices, and further experimentation, for example, using constraint hardware like smartphones for proving.

REFERENCES

[1] J. Groth, "On the size of pairing-based non-interactive arguments," in *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*. Springer, 2016, pp. 305–326.

[2] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. Ward, "Marlin: Preprocessing zksnarks with universal and updatable srs," in *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I 39*. Springer, 2020, pp. 738–768.

[3] M. Peise, J. Kuhlenkamp, A. Busse, J. Eberhardt, M.-R. Ulbricht, S. Tai, J. Baus, M. Kassebaum, and T. Zörner, "Blockchain-based local energy grids: advanced use cases and architectural considerations," in *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 2021, pp. 130–137.

[4] J. Eberhardt, M. Peise, D.-H. Kim, and S. Tai, "Privacy-preserving netting in local energy grids," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2020, pp. 1–9.

[5] T. Sund, C. Lööf, S. Nadjm-Tehrani, and M. Asplund, "Blockchain-based event processing in supply chains—a case study at ikea," *Robotics and Computer-Integrated Manufacturing*, vol. 65, p. 101971, 2020.

[6] J. Heiss, T. Oegel, M. Shakeri, and S. Tai, "Verifiable carbon accounting in supply chains," *IEEE Transactions on Services Computing*, 2023.

[7] J. Heiss, E. Grünewald, S. Tai, N. Haimerl, and S. Schulte, "Advancing blockchain-based federated learning through verifiable off-chain computations," in *2022 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 2022, pp. 194–201.

[8] C. Lee, J. Heiss, S. Tai, and J. W.-K. Hong, "End-to-end verifiable decentralized federated learning," *arXiv preprint arXiv:2404.12623*, 2024.

[9] M. C. Ballandies, H. Wang, A. C. C. Law, J. C. Yang, C. Gösken, and M. Andrew, "A taxonomy for blockchain-based decentralized physical infrastructure networks (depin)," *arXiv preprint arXiv:2309.16707*, 2023.

[10] "Verifiable credentials data model v1.1," 2021, https://w3.org/TR/vc-data-model/.

[11] "Verifiable credentials data model v1.1," Accessed on May 22, 2024, https://w3.org/TR/vc-data-model/.

[12] "Streamr: Decentralized data broadcasting," Accessed on May 22, 2024, https://streamr.network/whitepapers.

[13] "Accurast," Accessed on May 22, 2024, https://acurast.com/.

[14] J. Heiss, J. Eberhardt, and S. Tai, "From oracles to trustworthy data on-chaining systems," in *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 2019, pp. 496–503.

[15] J. Park, H. Kim, G. Kim, and J. Ryou, "Smart contract data feed framework for privacy-preserving oracle system on blockchain," *Computers*, vol. 10, no. 1, p. 7, 2020.

[16] Z. Wan, Y. Zhou, and K. Ren, "zk-authfeed: Protecting data feed to smart contracts with authenticated zero knowledge proof," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 2, pp. 1335–1347, 2022.

[17] J. Heiss, "Trustworthy data provisioning in blockchain-based decentralized applications," Ph.D. dissertation, Technische Universität Berlin, 2023.

[18] J. Heiss, A. Busse, and S. Tai, "Trustworthy pre-processing of sensor data in data on-chaining workflows for blockchain-based iot applications," in *Service-Oriented Computing: 19th International Conference, ICSOC 2021, Virtual Event, November 22–25, 2021, Proceedings 19*. Springer, 2021, pp. 133–149.

[19] M. Sporny, D. Longley, and D. Chadwick, "Verifiable credentials data model v1.1," 2021, https://w3.org/TR/vc-data-model/.

[20] J. Eberhardt and S. Tai, "Zokrates-scalable privacy-preserving off-chain computations," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018, pp. 1084–1091.

[21] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[22] D. Longley, M. Varley, and D. Zagidulin, "Verifiable presentation request v0.2," 2021, https://w3c-ccg.github.io/vp-request-spec/.

[23] J. Eberhardt and J. Heiss, "Off-chaining models and approaches to off-chain computations," in *Proceedings of the 2nd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*, 2018, pp. 7–12.

[24] J. Heiss, R. Muth, F. Pallas, and S. Tai, "Non-disclosing credential on-chaining for blockchain-based decentralized applications," in *International Conference on Service-Oriented Computing*. Springer, 2022, pp. 351–368.

[25] T. Dayaratne, X. Fan, Y. Liu, and C. Rudolph, "Ssi4iot: Unlocking the potential of iot tailored self-sovereign identity," *arXiv preprint arXiv:2405.02476*, 2024.

[26] E. Ben-Sasson, A. Chiesa, M. Green, E. Tromer, and M. Virza, "Secure sampling of public parameters for succinct zero knowledge proofs," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 287–304.

[27] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," *Cryptology ePrint Archive*, 2018.

[28] F. R. Vidal, F. Gouveia, and C. Soares, "Revocation mechanisms for academic certificates stored on a blockchain," in *2020 15th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE, 2020, pp. 1–6.

[29] G. Fedrecheski, J. M. Rabaey, L. C. Costa, P. C. C. Ccori, W. T. Pereira, and M. K. Zuffo, "Self-sovereign identity for iot environments: a perspective," in *2020 Global Internet of Things Summit (GIoTS)*. IEEE, 2020, pp. 1–6.

[30] S. K. Gebresilassie, J. Rafferty, P. Morrow, L. Chen, M. Abu-Tair, and Z. Cui, "Distributed, secure, self-sovereign identity for iot devices. 2020 ieee 6th world forum on internet of things (wf-iot), 1–6," 2020.

[31] P. C. Bartolomeu, E. Vieira, S. M. Hosseini, and J. Ferreira, "Self-sovereign identity: Use-cases, technologies, and challenges for industrial iot," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 1173–1180.

[32] X. Fan, Q. Chai, L. Xu, and D. Guo, "Diam-iot: A decentralized identity and access management framework for internet of things," in *Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure*, 2020, pp. 186–191.

[33] M. Luecking, C. Fries, R. Lamberti, and W. Stork, "Decentralized identity and trust management framework for internet of things," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2020, pp. 1–9.

[34] D. Yoon, S. Moon, K. Park, and S. Noh, "Blockchain-based personal data trading system using decentralized identifiers and verifiable credentials," in *2021 International Conference on Information and Communication Technology Convergence (ICTC)*, 2021, pp. 150–154.

[35] Privado.ID, "Introduction to privado id (prev. polygon id)," 2024, https://docs.privado.id/docs/introduction/.

[36] Hyperledger Foundation, "Hyperledger anoncreds," https://www.hyperledger.org/projects/anoncreds.

[37] J. Yin, Y. Xiao, Q. Pei, Y. Ju, L. Liu, M. Xiao, and C. Wu, "Smartdid: a novel privacy-preserving identity based on blockchain for iot," *IEEE Internet of Things Journal*, vol. 10, no. 8, pp. 6718–6732, 2022.

[38] R. Muth, T. Galal, J. Heiss, and F. Tschorsch, "Towards smart contract-based verification of anonymous credentials," in *International Conference on Financial Cryptography and Data Security*. Springer, 2022, pp. 481–498.