RACE for 2030
RELIABLE AFFORDABLE CLEAN ENERGY

Identification, Key Management and Trust for Inverters for Distributed Energy Resources

Final Report

Australian Government
Department of Industry, Science and Resources

AusIndustry
Cooperative Research Centres Program

RACE for Networks

Research theme N2: LV network and DER hosting capacity

ISBN: 978-1-922746-51-1

Industry Report

Identification, Key Management and Trust for Inverters for Distributed Energy Resources

May 2024

Project team

Monash University

- Prof. Carsten Rudolph
- Dr. Jiangshan Yu
- Dr. Thusitha Dayaratne

Selectronic

- David Shirley
- Tom Shirley

Citations

Dayaratne, T., Rudolph, C., Yu, J., Shirley, D. and Shirley, T. (2024). Identification, Key Management and Trust for Inverters for Distributed Energy Resources.

Prepared for RACE for 2030 CRC.

Project partners

MONASH University

selectronic

Acknowledgements

The authors would like to thank the stakeholders involved in the development of this report, in particular the industry reference group members (AEMO and DEIP Cybersecurity Group) who have given so generously of their time. Whilst their input is very much appreciated, any views expressed here are the responsibility of the authors alone.

Acknowledgement of Country

The authors of this report would like to respectfully acknowledge the Traditional Owners of the ancestral lands throughout Australia and their connection to land, sea and community. We recognise their continuing connection to the land, waters and culture and pay our respects to them, their cultures and to their Elders past, present and emerging.

What is RACE for 2030?

RACE for 2030 CRC is a 10-year co-operative research program with AUD350 million of resources to fund research towards a reliable, affordable, and clean energy future. racefor2030.com.au

Executive Summary

All new inverters in Australia and New Zealand have been mandated to possess internet connectivity to comply with the Australian standard AS4777.2:2020 since December 2021. This connectivity significantly expands the cybersecurity attack surface for inverters and exposes them to potential threats from criminal activities, professional hackers, opportunistic or automated hacks and even nation-state-controlled activities. Consequently, it becomes imperative to continually assess the risks associated with the compromise of these devices while ensuring the integrity and accuracy of the data received from these inverters. This is vital for the efficient and effective operation of grids that incorporate distributed energy resources.

The knowledge possessed by network operators about inverters is often inadequate for accurately estimating these risks. Inverters operate outside the security perimeter of network operators. Although encryption and authentication can safeguard against communication-based attacks, they provide limited defence against threats that exploit compromised inverters. Therefore, this project proposes the implementation of additional methods, including reliable identities, up-to-date software/firmware versions, and security metadata. These methods aim to facilitate more precise risk assessments and ultimately enhance the trustworthiness of inverters. The objective of this project is to develop a distributed framework for device identity management and status information, empowering network operators and virtual power plant (VPP) operators to establish secure communication and maintain ongoing compliance with security requirements.

This project addresses these challenges with a multifaceted approach. First, it introduces a novel methodology centred on verifiable credentials (VCs) to enhance the security of firmware update processes and track the history of firmware updates. This allows for more rigorous evaluation of trustworthiness of the inverters.

Furthermore, a unique trust-cycle is proposed to facilitate secure interactions between smart inverters and VPP operators. This innovative framework leverages manufacturer issued VCs that certify inverter firmware updates, enabling stakeholders to ensure that high risk devices are restricted with respect to grid operations.

In addition, the project proposes a VC-based mechanism to validate inverter capabilities and compliance with configurations, ensuring that inverters operate within the expected parameters. By bridging the gap between existing standards and evolving cybersecurity needs, the project highlights potential vulnerabilities within the IEEE 2030.5-2018 standard and offers enhanced architectures to mitigate these concerns.

Lastly, the project explores the untapped potential of defining VCs for common inverter use cases, further enhancing the security and functionality of these devices. In summary, this project presents a comprehensive framework that leverages cutting-edge technologies, such as VC, to enhance the security, compliance, and trustworthiness of inverters in the modern grid ecosystem.

Contents

1 Introduction

The global energy landscape is undergoing a significant transformation, with solar photovoltaic (PV) emerging as the most rapidly expanding form of energy generation. This shift toward sustainability is evidenced by solar PV contributing to approximately 10% of Australia's electricity generation and 5% in Europe. Notably, Australia alone boasts a cumulative solar PV capacity exceeding 11 GW, as reported by ARENA [1]. This remarkable proliferation of distributed energy resources (DERs) presents a promising avenue for the advancement of eco-friendly energy sources that meet the UN climate targets.

Distributed system operators (DSOs) are taking advantage of the concept of virtual power plants (VPPs) to effectively harness the potential of these dispersed energy generators. By capitalising on the willingness of consumers to share their distributed energy resources (DERs), operators aggregate consumer-owned solar PV installations, offering financial incentives to contributing consumers. More specifically, a VPP functions as a logical generation entity that aggregates DERs within a designated geographical area [2]. The VPP is governed and managed by an independent entity, commonly referred to as a VPP operator. This strategic arrangement empowers the seamless coordination and optimisation of DERs, paving the way for a more sustainable and efficient energy future. Several active VPP deployments are underway in Australia, including the South Australia (SA) VPP, the AGL VPP, Energy Australia's 'POWERRESPONSE' VPP, and the Ausgrid VPP.

VPPs are required to exchange information between different entities, including smart inverters, smart meters, home energy management systems, smart appliances, and electric vehicles. Despite the criticalness of these devices for optimising energy distribution, they operate outside the direct control of distribution network operators and utilities. However, operators must establish communication with these devices to ensure the reliability of grid operations. This distributed configuration significantly expands the potential attack surface, leaving the system vulnerable to threats from individuals to state-sponsored actors. Depending on the nature of the attack, the consequences can range from unauthorised financial gains to catastrophic grid failures. Therefore, ensuring end-to-end security within these setups is imperative.

The primary objective of this project is to thoroughly investigate and improve the secure integration of smart inverters within VPPs. This improvement includes aspects such as device identification, configuration management, and data integrity assurance. Achieving these security goals is imperative not only for VPPs, but also for the broader context of distribution networks and end-user equipment. To address these challenges, the project leverages decentralised identity and blockchain technologies, which hold the promise of improving security, transparency, and trust in energy systems. By focusing on these critical aspects, the project aims to improve the resilience and reliability of smart grid operations.

1.1 Preliminaries

1.1.1 Device Identity

The identity of an inverter plays a pivotal role in any smart grid application. Security-wise, identity stands as a paramount factor, as it aids in establishing trust both in the inverter and its transmitted data. Possessing a secure and reliable identity actively mitigates security concerns, as it empowers controllers to enforce robust access controls and implement appropriate monitoring measures.

The perfect identity should encapsulate distinct hardware and software attributes that are inherently unforgeable. These attributes firmly anchor the identity to the device itself, fostering a strong linkage. However, universally adopting such an identity across a large-scale or global context becomes impractical due to factors like the inherent diversity of inverter capabilities and the associated costs. Consequently, alternative methodologies are necessary to establish robust inverter identities, without being confined to physical attributes.

Several device identifiers exist, each suited to distinct contexts. Examples include serial numbers, IP addresses, MAC addresses, or manufacturer digital certificates. Widely adopted standards like CSIP and IEEE 2030.5-2018 define inverter identities based on certificates issued by manufacturers. Specifically, these standards outline the following identities:

1. Short form device identifier (SFDI) - Left-truncated first 36 bits of the device certificate fingerprint (SHA256 hash) expressed as 12 decimal digits.
2. Long form device identifier (LFDI) - Left-truncated first 160 bits (20 bytes) of the device certificate fingerprint (SHA256 hash) expressed as 40 hexadecimal (base 16) digits in groups of four.
3. 6-digit PIN

Certificates link the device to a cryptographic identity. While this can be considered more secure than serial numbers or MAC addresses, a compromised device can still provide wrong information about its state, and cryptographic keys can be extracted. In principle, additional secure hardware, such as a Trusted Platform Module, can provide a secure devices identity and remote attestation to ensure that the device, even if compromised on software level, cannot lie about itself. The problem is that inverters generally do not run with enabled trusted hardware security.

1.1.2 Device Firmware

Device firmware plays a pivotal role in the functionality, security, and overall performance of intelligent electronic devices. It serves as the software that operates the device's hardware components and defines how the device interacts with its environment and users. The significance of firmware extends beyond mere functionality; it encompasses critical aspects such as security, stability, and feature enhancements. Firmware updates are frequently released by manufacturers to address security vulnerabilities, bugs, and performance improvements. These updates are essential for ensuring the device's resilience against evolving threats. Outdated or vulnerable firmware can expose devices to cyberattacks, potentially compromising user data and network security. Consequently, maintaining up-to-date and secure firmware is not just a matter of functionality but a fundamental requirement for the longevity and integrity of electronic devices.

1.1.3 Device Information

Device information plays a pivotal role in various smart grid operations. Access to detailed information about the inverter and associated DER allows stakeholders, such as VPP operators, utilities, and regulatory bodies, to effectively oversee and manage DER assets. Moreover, these stakeholders can precisely evaluate the inherent risks associated with participating or communicating inverters, ensuring the security of operational processes and enabling timely implementation of required actions and countermeasures. Nevertheless, it is difficult to obtain desired and trustable device information given that inverters/DERs reside outside utility companies' control perimeter. Hence, the widely used IEEE 2030.5-2018 standard defines a set of commonly device attributes under the '*DeviceInformation*' package depicted in Table 1.

Table 1. Attributes defined in 'DeviceInformation' package of IEEE 2030.5-2018.

Attribute	Description
GPSLocationType	GPS location of this device
LFDI	Long form device identifier
mfDate	Manufactured date and time
mfHwVer	Manufacturer hardware version
mfID	Manufacturer's IANA Enterprise Number
mfInfo	Manufacturer dependent information related to the manufacture of this device
mfModel	Model number
mfSerNum	Serial number
primaryPower	Primary source of power
secondaryPower	Secondary source of power
swActTime	Activation date/time of currently running software
swVer	Currently running software version
DRLCCapabilities	Static capabilities of the device

1.1.4 Decentralised Identifiers (DID)

A DID is a globally unique and persistent identifier. DIDs enable verifiable and decentralised digital identities for entities [3] that are under total control of the identity owner, instead of relying on a central authority. The W3C approved the DID specification as a recommendation in late June 2022. The DID representation conforms with the URI (universal resource identifier) scheme with three mandatory components as below.

did:<did-method>:<method-specific-identifier>

A DID resolves to a DID document, which is a JSON-LD (JavaScript Object Notation-Linked Data) object. The DID document possesses public keys for cryptographic operations, including authentication and verification.

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2018/v1"
  ],
  "id": "did:sov:123456789",
  ...
  "verificationMethod": [{
    "type": "Ed25519VerificationKey2018",
    "id": "did:sov:23456789#key-1",
    "publicKeyBase58": "......"
  }],
  "authentication": [
    "did:sov:123456789#key-1"
  ],
  "service": [{
    "id": "did:sov:123456789#mqtt",
    "serviceEndpoint": "broker.example.mqtt.com"
  }]
}
```

Listing 1: Sample DID document for the manufacturer

The DID document can also contain service endpoints, which can be used to advertise different services provided by the entity or the controller of the entity. Listing 1 depicts a sample DID document.

Figure 1 presents an overview of the DID architecture. A DID can reference various entities, including both physical and logical entities. The DID document outlines the associated keys and endpoints of the referenced entity. DID documents are typically stored in publicly accessible verifiable data registries such as blockchains or distributed file systems like IPFS. In most scenarios, DID documents should be publicly accessible to facilitate communication initiation between stakeholders and the entity, similar to how digital certificates function in a PKI infrastructure. The DID itself is presumed to remain immutable once created. Nevertheless, the keys and endpoints associated with the DID can be updated within the DID document. By default, the entity is the owner/controller of the DID document, thereby having full control over its updates. However, the *controller* property can override this default, allowing separate controllers to be designated for the DID document. For instance, in the context of inverters, an inverter might lack the capability to manage its own DID document. In such cases, the inverter owner takes control of its DID document. This behaviour is still expected and common in smart grid use cases. AEMO's project EDGE [4] stands as the pioneering experimental project that introduced the DID concept into the Australian grid context.

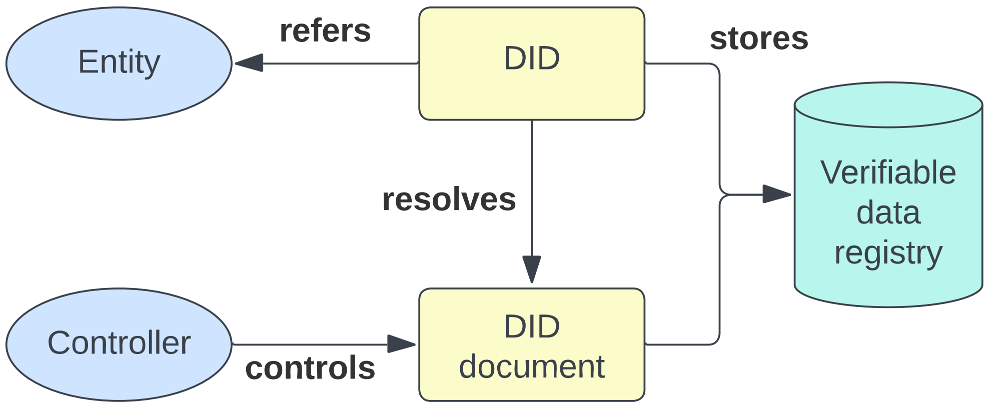

Figure 1: High level view of the DID architecture

```
{
  ″@context″: [
    ″https://www.w3.org/2018/credentials/v1″,
    ..
  ],
  ″id″: ″did:iot:vc:123456789″,
  ″type″: [″VerifiableCredential″, ″IoTVC″],
  ″issuer″: ″did:iot:manufacturer:123456789″,
  ″validFrom″: ″2023-08-01T10:11:12Z″,
  ″credentialSubject″: {
    ″owner″: ″did:iot:user:123456789″,
    ″serialNo″ : ″1234567899″,
    ..
  },
  ″credentialSchema″: {
    ..
  },
  ″credentialStatus″: {
    ..
  },
  ″proof″: {
    ..
  }
  ..
```

Listing 2: Structure of a sample VC

1.1.5 Verifiable Credential (VC)

A VC represents statements made by an issuer in a tamper-evident and privacy-respecting manner [5]. Specifically, VCs provide portable and provable claims about an entity that can be cryptographically verified.

The W3C maintains the VC specifications, which consist of three components: Metadata, claims, and proof as shown in Listing 2. Metadata describes meta-information about the VC, such as the issuer, expiration date, and public keys for verification. It also includes a set of claims about the entity and proofs.

The VC ecosystem is depicted in Figure 2. It consists of three main entities (issuer, holder and verifier) and a supporting verifiable data registry. Any entity can be a VC issuer. However, the trustworthiness of a particular VC depends on the issuer. Thus, VCs are usually issued by reputed or trusted entities in practical contexts. The holder stores their VCs in a digital wallet and presents them to the verifier upon request. Verifiers can verify the presented VCs without contacting the issuer, as verification keys can be accessed via the verifiable data registry. This prevents issuers from correlating the holder and the services that they access.

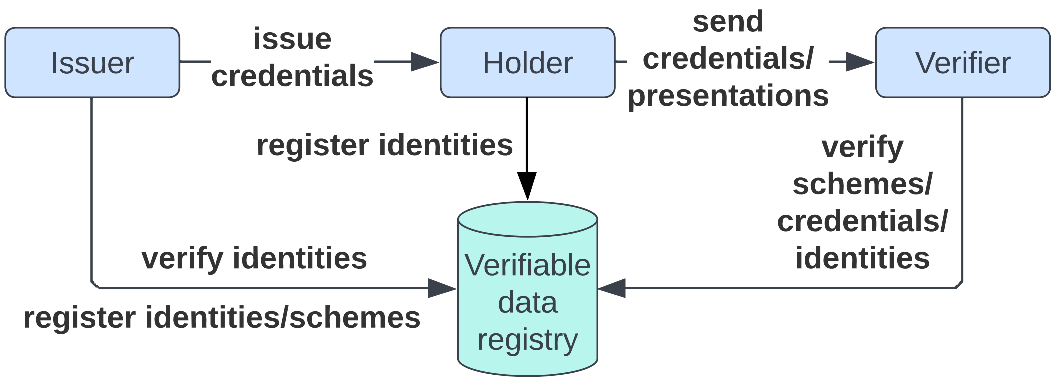
Issuer
issue credentials
Holder
send credentials/ presentations
Verifier
register identities
verify schemes/ credentials/ identities
verify identities
Verifiable data registry
register identities/schemes

Figure 2: High level architecture of the verifiable credentials

1.1.6 Remote Attestation (RA)

Remote attestation (RA) is a fundamental security service that allows one system to verify the integrity and security of another remote system or device. It plays an essential role in distributed setups where no single entity has full control over the entire system, yet the entire system relies on communication and collaboration.

In RA, a requesting system, often referred to as a verifier, seeks confirmation that the remote system, known as the prover, is in the expected state. This expected state can include information about its software version, hardware configuration, and the presence of security patches. The high-level abstraction of the RA process is depicted in Figure 3. The prover initiates the process by generating a challenge, which is then sent to the verifier. Various RA protocols employ different approaches, such as nonces, memory regions, or inputs, when creating the challenge. The prover constructs the attestation response based on the received challenge, typically specifying its internal state and memory content, and then sends the response to the verifier. The verifier, in turn, verifies the attestation response using pre-computed or expected values. RA is widely used in various domains, including securing IoT devices, enhancing cloud computing security, and ensuring the integrity of critical infrastructure components such as servers and routers.

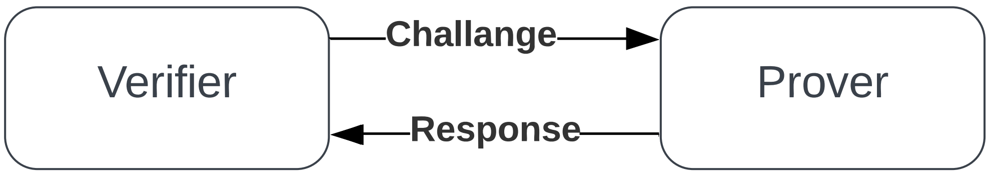

Figure 3: High-level flow of remote attestation

1.2 System Model

We analyse the use of smart inverters within a microgrid scenario, where these intelligent devices play a pivotal role in ensuring demand-supply balance as part of an aggregator/VPP setup. The interaction between the entities involved is shown in Figure 4. Individuals who own DERs, such as solar panels and battery storage units, procure smart inverters from manufacturers and integrate them into their setups. Each DER is connected to a smart inverter that manages, monitors, and regulates the associated DER. Although practical setups often include gateways and home energy management systems (HEMS), we simplify this by treating the combination of inverter and gateway/HEMS as a single entity. It is assumed that each smart inverter possesses a dependable static identifier that manufacturers assign during the manufacturing phase. These inverters are equipped with firmware to control the underlying hardware and software to facilitate interaction between the inverter and the user/operator. Manufacturers periodically release updated firmware/software versions to rectify bugs, address security vulnerabilities, and enhance performance, as needed.

We assume that a group of consumers enrol their inverters into a VPP. Upon enrolment, the VPP operator sends control or monitoring signals to all enrolled inverters or a designated subset. These inverters authenticate the legitimacy of the commands and execute them. Subsequently, the requested data or the resulted status information are sent back to the VPP operator. The operator then validates these responses, assessing the condition of each device in relation to the broader grid and the VPP condition.

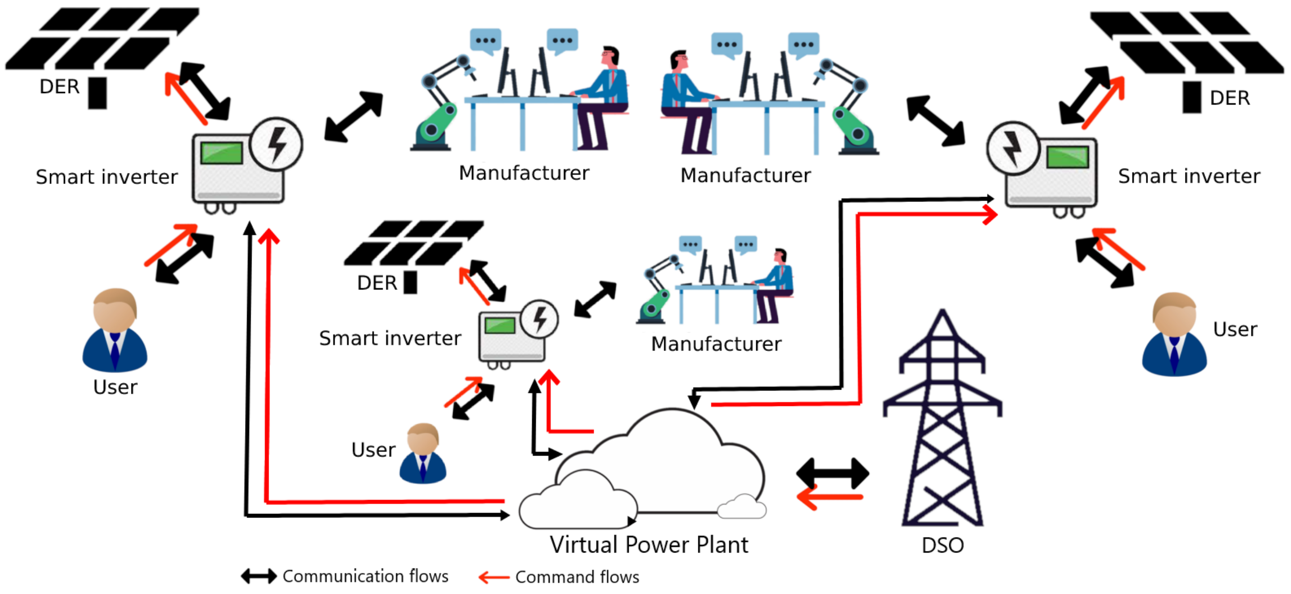

Figure 4: Entities and their interaction flows

1.3 Setup

General public does not have any involvement over the use case. Therefore, the use of a public blockchain is unnecessary. On the contrary, a private blockchain is unsuitable due to the existence of multiple stakeholders. Hence, we propose using a consortium blockchain in the design, as it provides the correct amount of security and decentralisation. The DID API is assumed to manage DIDs, including generating DIDs and retrieving existing DID documents. DID API further facilitate the DID document updates. Within the context of a consortium blockchain, we assume that each user possesses a key pair to interact with the chain. Additionally, we assume the followings throughout this work, which are explained in detail in appropriate sections.

1. Inverter manufacturers possess public DIDs
2. Each smart inverter is assigned a public DID by its manufacturer
3. Smart inverters possess a digital wallet to store keys and VCs
4. Manufacturers register DIDs and VC schemes on a suitable blockchain platform
5. There is a secure communication channel between the inverter and manufacturer
6. Manufacturers possess a mechanism to determine the firmware version of an inverter and its installed datetime
7. Successful secure booting of inverters

All these assumptions can be realised with software/firmware solutions. Thus, in principle, no additional hardware components are required and there is no significant cost increase for manufacturers. However, higher security levels can potentially be achieved by integrating secure hardware components such as trusted platform modules.

2 Use case 1: Tracking Smart Inverter Firmware Updates for Secure VPP Operations

2.1 Introduction

The integration of smart inverters, serving as the interface for DERs, plays a pivotal role in modern smart grid operations. Consequently, ensuring the trustworthiness of smart inverters stands as a fundamental requirement for safeguarding the security and reliability of grid operations. Distributed setups, wide-area communication, and remote control/sensing between smart inverters and utilities/operators significantly expand the attack surface for cyberattacks. More importantly, existing standards and protocols have overlooked some of the critical aspects of cybersecurity. Furthermore, adversaries can bypass existing security measures by leveraging software-based attacks in compromising firmware / software of the equipment to gain access. Therefore, having a reliable firmware update process is essential to ensure the security of grid operations and that inverters consistently run firmware versions without known security vulnerabilities.

Despite the paramount importance of maintaining up-to-date firmware and software versions in smart inverters, relatively little attention has been devoted to this in grid context. We address this critical gap by introducing a novel framework that harnesses the power of verifiable credentials to effectively manage and monitor firmware updates in smart inverters. Furthermore, we introduce a trust cycle rooted in firmware updates, designed to enhance the security of interactions between smart inverters and various stakeholders within the smart grid ecosystem.

2.2 Tracking Inverter Firmware Update

This section introduced the VC-based security model that improves the security of VPP operations by tracking inverter firmware updates. The entire process from update manufacturing to report firmware updates is depicted in Figure 5. We divide the process into multiple sub-processes for brevity.

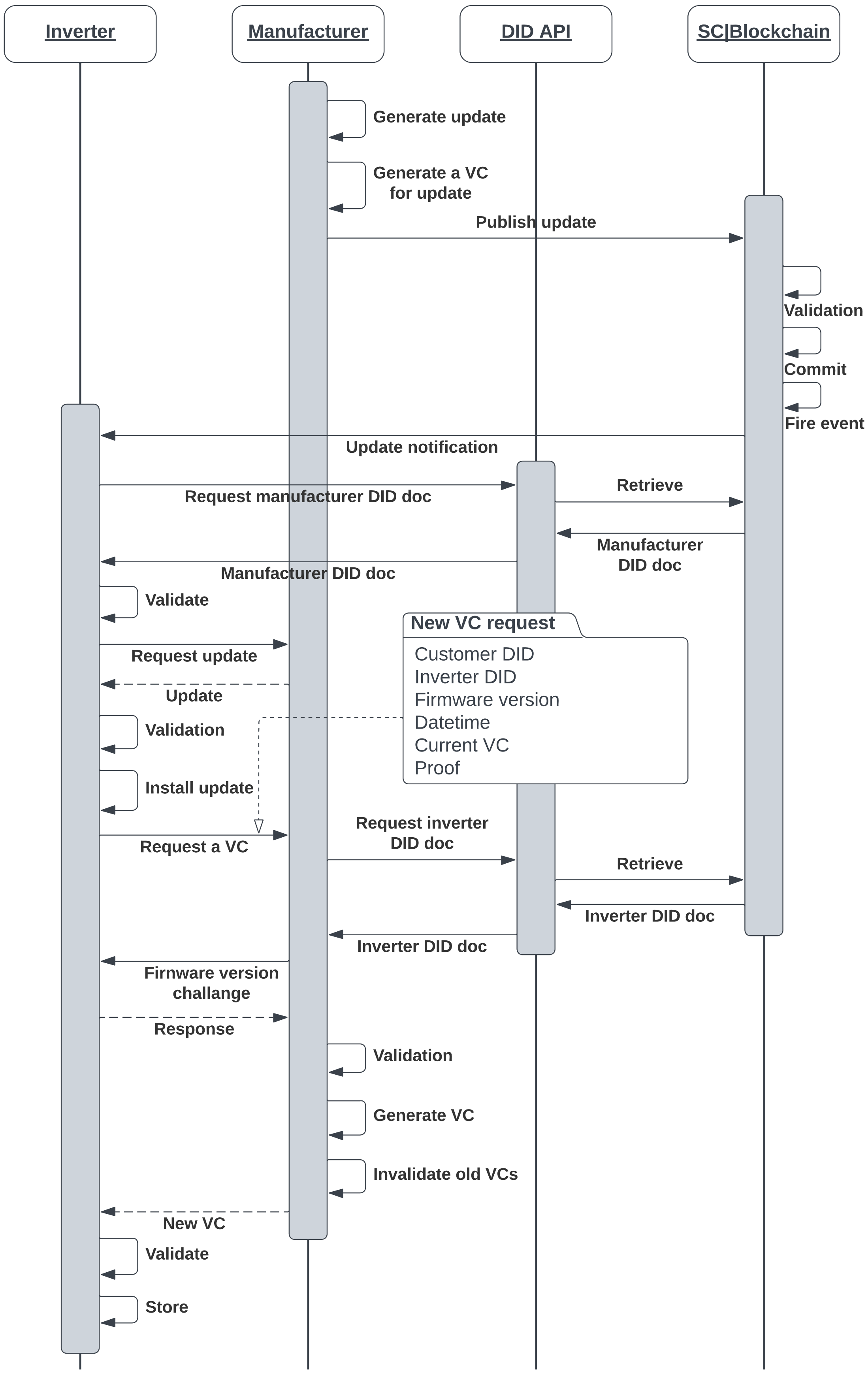

Figure 5: Sequence diagram for the firmware update process

2.2.1 Firmware Update Management Smart Contract

Given the existence of the blockchain infrastructure, the manufacturer deploys a smart contract into the blockchain to manage the availability of its firmware updates. This smart contract emits events whenever a new update is made available. The smart contract should incorporate suitable validation mechanisms to ensure that only the manufacturer has the authorisation to introduce new update information. This measure prevents malicious users within the system from injecting false update data into the ecosystem. A sample smart contract is shown in Listing 3.

```
contract ManufacturerUpdate {

        address private manufacturer;

        constructor() {

                manufacturer = msg.sender;

        }

        event NewUpdate(address indexed from, string version, string model, string credential);

        function SaveUpdate(address _to, string _version, string _model, string _credential) public {

                require(manufacturer == msg.sender, "Only manufacturer can push updates");

                // Save update VC logic here

                emit NewUpdate(msg.sender, _version, _model, _credential);

        }

}
```

Listing 3: Sample smart contract

2.2.2 Inverter Manufacturing

Upon the manufacturing of an inverter, the manufacturer assigns a key pair consisting of a public and a private key, along with a public DID generated using the DID API for the inverter. The inverter's DID document holds the public key of the inverter. This DID document is stored on the blockchain or on a distributed file system, such as IPFS, based on the DID API implementation. Once the DID document is published, the manufacturer embeds the inverter's key pair, DID, and the manufacturer's DID into the inverter. We assume the inbuilt inverter wallet is capable of storing this information. Furthermore, the manufacturer subscribes the inverter to the smart contract events deployed on the blockchain. Established Web3 providers, such as MetaMask, in combination with libraries such as Web3.js/ether.js, can be employed to effectively listen and subscribe to these events. This functionality can be seamlessly integrated into the inverter's software.

2.2.3 Publishing New Updates

The manufacturer is responsible for generating the firmware update along with the corresponding VC for the update. The proposed structure of this VC is displayed in Listing 4. This VC encapsulates vital details concerning the update. Specifically, we suggest incorporating manufacturer's DID, release date, hyperlink to the update downloadable location, version number, hash value of the binary, and a list of supported inverter models within the VC. Additionally, we propose to include the CVE (Common Vulnerabilities and Exposures) numbers that are addressed from the update. Updates can be stored directly on the blockchain or within an update server.

```
{
 "@context":[..],
 "id": "did:sov:sg:inverter:firmware:vc:123456789",
 "type": ["VerifiableCredential", "FirmwareVC"],
 "issuer": "did:sov:sg:manufacturer:123456789",
 " validFrom": "2023-04-01T10:11:12Z",
 "credentialSubject": {
  "manufacturer": "did:sov:sg:manufacturer:123456789",
  "releasedDate": "2023-04-01T10:00:00Z",
  "link": "<url to download the binary>",
  "frimwareInfo": {
     "version": "1.0021",
     "binaryHash": "ARVDVX2753hd6752H",
  },
  "associatedCVEs": {
     //list of CVE numberd that are addressed from the update
  },
  "supportingModels":{
     //all the model numbers that supports the firmware
  }
 },
 "credentialSchema": {..},
 "credentialStatus": {..},
 ...
 "proof": {..}
}
```

Listing 4: Sample VC for an update

Upon generation of VC, the constructed VC and the public DID of the manufacturer are sent to the smart contract (SC) deployed on the blockchain. The manufacturer's private key (SK) corresponding to the DID is used to sign the message to ensure integrity and authenticity. The smart contract verifies the message signature using the manufacturer's public key (PK) specified in the corresponding DID document (DIDDoc) or associated with the manufacturer's blockchain account. Once the verification is a success, the VC is committed to the blockchain, which triggers a smart contract event to notify the availability of a new firmware version.

2.2.4 Firmware Update Process

Once the inverter is installed in a home, the firmware needs to be updated whenever required. Inverters (SI) get notified of the availability of new updates, as inverters are listening to the events of the corresponding manufacturer's smart contract. Upon receiving an event, the inverter checks if the model matches and if the version is newer than the current version of the inverter. If both conditions are satisfied, the inverter saves the received VC for the update on its wallet. The inverter can directly refer to the download link specified in the VC to download the update. In scenarios where the link does not exist on the VC, the inverter requests the manufacturer's DID document from the DID API using the embedded manufacturer DID. The DID document lists the manufacturer's update endpoint. Inverter extracts the endpoint from the document and requests the update from the update server. Upon receiving the update, the inverter verifies the integrity of the update by comparing the received update hash with the update hash (*binaryHash*) stated in the VC received from the smart contract. The inverter installs the update if the verification succeeds or discards the update if it fails. We emphasise the assumption on secure bootstrapping of the inverter as it is pivotal in establishing the root-of-trust for the firmware update process.

2.2.5 Issue a New VC for the Inverter

A successful update updates the firmware version of the inverter. Nevertheless, the newer version will not automatically reflect on the firmware version stated in the inverter's VC, which is issued by the manufacturer. Thus, an updated VC should be generated, and the current VC should be invalidated to reflect the changes in the update history.

The user requests a new VC from the manufacturer using the VC endpoint listed in the manufacturer's DID document. The request consists of the customer DID, inverter DID, firmware version, installed date-time, current VC, and nonce. Hardware-based RA is the best solution to obtain the installed date and time of the firmware and the version. However, we assume that the manufacturer embeds a software-based method (software-based RA) to determine the firmware version of an inverter and its installed date-time given the unavailability of hardware-based RA method. A potential method to implement the update-specific string is the use of an update-specific, one-time password approach, similar to the time-based one-time password TOTP method. Users use the update-specific string (P) generated from the update instead of manually specifying the update version and date-time. The request should be digitally signed by the user's private key corresponding to the public DID of the user. Manufacturer's public key can be used to encrypt the signed message to ensure the integrity and privacy of inverter details.

```
{
 ″@context″:[″https://www.w3.org/2018/credentials/v1″,
       ″https://example.com/2021/credentials/inverters/v1″],
 ″id″: ″did:sov:sg:inverter:vc:123456789″,
 ″type″: [″VerifiableCredential″, ″InverterVC″],
 ″issuer″: ″did:sov:sg:manufacturer:123456789″,
 ″ validFrom″: ″2023-04-01T10:11:12Z″,
 ″credentialSubject″: {
  ″immutable″: {
     ″id″: ″did:sov:sg:inverter:123456789″,
     ″serialNo″: ″123456789″,
     ″manufacturedDate″:″2021-01-01T00:01:02Z″,
     ..
  },
  ″updatable″: {
     ″owner″: ″did:sov:sg:user:123456789″,
     ″status″ : ″active″,
     ″softwareVersion″: ″v2.0″,
     ″timelyUpdated″:true,
     ″missingUpdates″:false,
     ..
  },
  ″firmwareHistory″: {      ″v2.0″: ″2023-04-01T08:11:12Z″,
                            ″v1.8″: ″2022-11-17T16:34:20Z″},
  ″resetHistory″:{      //factory reset date-times     }
 },
 ″credentialSchema″: {..},
 ″credentialStatus″: {..},
 ...
 ″proof″: {..}
}
```
Listing 5: Sample VC for an inverter

The manufacturer then validates the update specific value to ensure the version and installed date and time. Upon successful validation, the manufacturer generates the VC reflecting the changes in the update history once ownership is verified. It also invalidates any previously issued VC for the inverter. The new VC changes the current '*softwareVersion*' attribute in the VC and appends the version and the datetime to the '*firmwareHistory*' section, accordingly. A sample inverter VC is depicted in Listing 5. The manufacturer signs the VC and sends it to the inverter. The inverter stores the VC in its wallet once it verified that the issuer is the manufacturer. Detailed communication protocol and interactions among entities are depicted in Algorithm 1.

The manufacturer decrypts the message using its private key and then proceeds to validate the authenticity of the message using the public key listed in the user's DID document. Furthermore, it verifies the ownership of the inverter by referring to the `controller' property stated in the inverter's DID document or the current VC issued for the user, thereby confirming ownership. Failure to satisfy either of these conditions results in the manufacturer rejecting the issuance of a new VC.

The next step involves the manufacturer validating the update-specific values to ensure accuracy regarding the version and installation date and time. Upon successful validation, the manufacturer generates a new VC to reflect the changes in the update history. The manufacturer also invalidates any previously issued VC for the inverter. The new VC includes modifications to the current '*softwareVersion*' attribute in the VC and appends the version and datetime details to the '*firmwareHistory*' section as required. A sample inverter VC is provided in Listing 5. Subsequently, the manufacturer signs the VC, encrypts the signed VC using the user/inverter's public key, and transmits it to the inverter/user. The user/inverter decrypts the message and ensures the authenticity of the message. The new VC will be stored in the inverter's wallet after verifying the issuer's authenticity. Additionally, previously issued VCs need to be removed from the wallet. Detailed communication protocols and interactions among entities are depicted in Algorithm 1.

Algorithm 1: Firmware Update

```
$M$              Generate a firmware version ($FW^i_{M_i}$)
                 Publish the firmware to the update server/cloud
                 Generate a new VC ($VC^{FW_i}_{M_i}$) for the firmware
$M \to SC$       Send the new update request
                   $msg = VC^{FW_i}_{M_i} || DID^P_{M_i} || nonce_1$
                   $sig = sign(hash(msg), SK^{DID}_{M_i})$
                   $newUpdReq = (msg||sig)$
$SC$             Verify the signature and request using $PK^{DID}_{M_i}$
                 Store the $VC^{FW_i}_{M_i}$ on blockchain/db/ipfs
                 Fire update available event
$SI$             Listen to update events
                 Receive update event
$SI \to M$       Request update ($FW^i_{M_i}$)
$M \to SI$       Send update ($FW^i_{M_i}$)
$SI$             Obtain $VC^{FW_i}_{M_i}$ from DID/VC-API/Wallet
                 Obtain $DIDDoc^P_{M_i}$ from DID/VC-API
                 Compare hash values
                   $H^{authentic}_{FW_i} \leftarrow$ Hash value stated on $VC^{FW_i}_{M_i}$
                   $H^{received}_{FW_i} \leftarrow hash(FW^i_{M_i})$
                   $H^{authentic}_{FW_i} == H^{received}_{FW^i_{M_i}}$
                 Verify signature
                   $sig \leftarrow signature(VC^{FW^i_{M_i}}_{M_i})$
                   $verify(sig, hash(VC^{FW^i_{M_i}}_{M_i}), PK(DIDDoc^P_{M_i}))$
                 Install $FW^i_{M_i}$
                   $proof\_value(P_i) \leftarrow$
                       $firmwareVerifier(FW^i_{M_i})||timestamp(t_i)$
$U \to M$        Send a new VC request
                   $msg = DID^P_{SI_iM_i}||DID^P_{U_i}||P_i||VC^{SI_i}_{M_i}||t_j||nonce_2$
                   $sig = sign(hash(msg), SK^{DID}_{U_i})$
                   $newVCReq = (msg||sig)$
                   $enryptVCReq = encrypt(newVCReq, PK^{DID}_{M_i})$
$M$              Decrypt the message using $SK^{DID}_{M_i}$
                 Verify the request using $PK^{DID}_{U_i}$
                 Generate a new VC ($VC^{SI_i}_{M_i}$) for the inverter
                 Invalidate old VCs
$M \to U$        Send the $VC^{SI_i}_{M_i}$
                   $msg = VC^{SI_i}_{M_i}||(nonce_2 + 1)$
                   $sig = sign(hash(msg), PK^{DID}_{M_i})$
                   $newVCRes = (msg||sig)$
                   $enryptVCRes = encrypt(newVCRes, PK^{DID}_{U_i})$
$U$              Decrypt the message using $SK^{DID}_{U_i}$
                 Verify the request using $PK^{DID}_{M_i}$
$U \to SI$       Upload $VC^{SI_i}_{M_i}$ to the inverter
$SI$             Check if the VC's issuer is manufacturer
                   $issuer(VC^{SI_i}_{M_i}) == DID^P_{M_i}$
                 Store the $VC^{SI_i}_{M_i}$
                 Remove old VCs
```

2.3 Leverage Inverter Trust for Secure VPP Operation

The VPP operator has no control or visibility over the firmware updates of the enrolled inverters, which makes the entire system vulnerable. Compromised inverters can provide inaccurate information (data/status) and potentially provides an entry point into the system. To address this vulnerability, we propose categorising inverters into different trust levels based on their firmware update history. Each trust level should have a well-defined set of interactions, with reduced interactions for lower trust levels. However, it is important to note that the trust state should not be a global attribute of the inverter itself, as different use cases may require different levels of trust. Instead, the trust state in the VPP context is defined by the VPP operator and only applicable within the specific VPP operator's operations.

The proposed trust cycle for inverters is illustrated in Figure 6. This cycle outlines three distinct trust levels: Trustable, Semi-Trust, and Distrust. These trust states evolve over the course of an inverter's lifecycle, primarily influenced by its firmware update history.

Initially, when an inverter leaves the manufacturer's facility, it is presumed to be in the `Trustable' state. This status remains intact if the inverter promptly updates its firmware upon the release of a new security updates or within a reasonable time-frame defined by the VPP operator. The determination of this reasonable time-frame is often contingent on the severity of security vulnerabilities associated with the previous firmware version.

If an inverter fails to update its firmware within the prescribed reasonable period, its trust state transitions to 'Semi-Trust.' This indicates that an inverter may be in the 'Semi-Trust' state even if it operates on the latest available firmware version, signifying a delay in updating to the most recent release.

The 'Distrust' state is assigned to an inverter that either remains without updates or runs a firmware version with known security vulnerabilities. An inverter in the 'Distrust' or 'Semi-Trust' state can only be restored to 'Trustable' status by undergoing a factory reset and receiving all available updates. Manufacturers are expected to incorporate a mechanism for determining the factory reset date and time.

Similarly to the process of installing a firmware update, users can request a new VC for the inverter in the case of a factory reset. The manufacturer modifies the existing VC to reflect the factory reset history, including the addition of the reset date and time to the '*resetHistory*' section in the VC (for a practical example, refer to Listing 5.

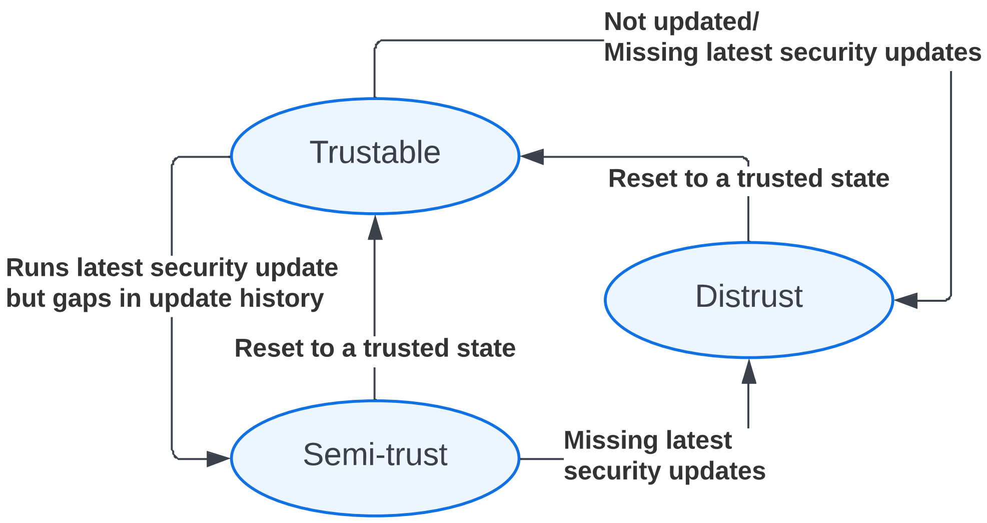

Figure 6: Proposed trust cycle for inverter based on update history

The proposed trust cycle can be implemented at the VPP operator, leveraging the '*timelyUpdated*' and '*missingUpdates*' attributes stated in the inverter VC. In this case, the manufacturer decides the threshold to determine if an inverter is updated within a reasonable time window. However, the VPP operator may need to tighten the time window or relax it depending on their use case and specific vulnerabilities. Thus, the operators should be able to possess their own implementation of the '*timelyUpdated*' attribute. The VPP operator can use the inverter's VCs and the list/VC of the available updates issued by the manufacturer. The manufacturer can provide published updates for a specific inverter model. A simple REST API can be implemented at the manufacturer to provide a list of updates for a given inverter model. The VPP operator can verify a specific inverter's update history (on the inverter's VC) against the available update list. If the inverter possesses the latest update and all the updates are installed within the predefined reasonable time window, the inverter is in the '*trustable*' state. The inverter's trust state is \`semi-trust' or \`distrust' if there are delays in installing updates or missing updates, as described. Algorithm 2 depicts the pseudocode to determine the trust state of an inverter.

Algorithm 2: Pseudo code for the trust state calculation

```
Input: inverter's VC ($VC^{SI_i}_{M_i}$), available update list and threashold $T$
Output: Trust state for the inverter
$trustState \leftarrow$ "distrust"
$timelyUpdated \leftarrow$ True
$allUpdates \leftarrow$ True
if $resetHistory$ exist in $VC^{SI_i}_{M_i}$ then
    $latestResetTime \leftarrow top(VC^{SI_i}_{M_i}["resetHistory"])$
    modify the updateList to only reflect updates published after the latestResetTime
end
for each availableUpdate($AvUpdt_i$), availableTime ($AvUpdt^t_i$) in the updateList do
    $missedUpdate \leftarrow True$
    for each installUpdate($InstlUpdt_i$), installTime ($InstlUpdt^t_i$) in the $VC^{SI_i}_{M_i}$ do
        if $AvUpdt_i == InstlUpdt_i$ then
            $missedUpdate \leftarrow False$
            $timelyUpdated \leftarrow timelyUpdated\ AND\ (AvUpdt^t_i + T > InstlUpdt^t_i)$
            break
        end
    end
    $allUpdates \leftarrow allUpdates\ AND\ !missedUpdate$
    if $missedUpdate == True$ then
        break
    end
end
if $allUpdates$ then
    if $timelyUpdated$ then
        $trustState \leftarrow$ "trustable"
    else
        $trustState \leftarrow$ "semi-trust"
    end
end
```

The VPP operator can fully utilise the data from the inverters in the `trustable' state in all operations/optimisations/calculations and issue authorised control signals. Data from the inverters in the '*semi-trust*' state can be consumed with some verification or uncertainty factors. However, the operator cannot ensure the reliable execution of control signals by these inverters. Operators should not use data from inverters in the `distrust' state for any operation, as the inverters in that state have a significant probability of being compromised.

2.4 Discussion

Although it is convenient to use third-party dependencies in developing the framework, it is essential to limit their use to ensure firmware security, as they can introduce additional vulnerabilities. Publishing VCs for updates can potentially raise concerns, as adversaries may learn about vulnerabilities in previous versions. Omitting CVE numbers from the update VC can address this issue. However, we believe that the association between the firmware version and the addressed CVE numbers should be accessible to authorised entities. Therefore, we propose introducing a separate VC to capture the associated CVE numbers, which would not be publicly accessible. Instead, it would only be shared with authorised stakeholders, such as DNSPs and VPP operators, upon request. A more robust approach would be to implement a zero-knowledge proof-based system that exposes an interface where interested stakeholders can interact to confirm whether a particular firmware addresses specific CVEs.

In this work, we assumed the existence of a secure channel between the manufacturer and the inverter for both communication and the update delivery process. Existing encryption-based secure communication mechanisms can be adopted for this purpose. However, we acknowledge that manufacturers can also implement a proprietary communication method, leveraging the availability of public key information in the DID documents.

Firmware installing process is independent from the use of VC. Thus, the proposed framework supports offline updates, where the owner can request the VC using the proof value. A public or consortium blockchain can be utilised in implementing the proposed solution. However, a permissioned blockchain (managed by manufacturers and VPP operators) is the better option, given the limited number of stakeholders and the unnecessary need for the public availability of the information. Existing DID and wallet implementations can be utilised in implementing the proposed framework instead of developing novel methods. Inverter owners should be able to request VCs from the manufacturer on behalf of the inverters, where the owner's identity can be linked from the same blockchain or other mechanisms. The firmware update process can be further secured by integrating an antivirus scan for the downloaded firmware.

The VPP operator can implement more granular levels of the trust cycle depending on their level of interactions. VCs can be stored on the blockchain directly. However, it can raise privacy issues. Thus, an inbuilt inverter or the owner wallet is the better choice to store VCs.

While it is convenient to utilise third-party dependencies during the framework's development, it is crucial to limit their use to maintain firmware security, as such dependencies can introduce potential vulnerabilities. The publication of VCs for updates could potentially raise concerns, as it may inadvertently disclose information about vulnerabilities present in previous firmware versions. To address this issue, the inclusion of CVE numbers in the update VC could be omitted. Nevertheless, we contend that the association between a firmware version and the corresponding CVE numbers should remain accessible to authorised entities. Hence, we propose the introduction of a separate VC designed explicitly to capture these associated CVE numbers, with the condition that it is not publicly accessible. Instead, access would be restricted to authorised stakeholders, such as DNSPs and VPP operators, and granted only upon request. A more robust approach could involve implementing a zero-knowledge proof-based system that provides an interface for interested stakeholders to interact with, confirming whether a particular firmware version addresses specific CVEs.

In our work, we assumed that a secure communication channel exists between the manufacturer and the inverter for both general communication and the delivery of firmware updates. While existing encryption-based secure communication mechanisms can be readily adopted for this purpose, we acknowledge that manufacturers also have the flexibility to implement proprietary communication methods, leveraging the public key information available in the DID documents.

The firmware installation process is independent of the use of VCs. Consequently, our proposed framework supports offline updates, where the owner can request the VC using the proof value (which validates the installed version and datetime). While the implementation of a public or consortium blockchain is feasible, we suggest that a permissioned blockchain managed by manufacturers and VPP operators is a more suitable choice, given the limited number of stakeholders and the absence of a necessity for public accessibility to the information. Rather than developing novel methods, existing DID and wallet implementations can be leveraged to implement the proposed framework. While VCs can indeed be stored directly on the blockchain, this may raise privacy concerns. As such, it may be more prudent to store VCs in an inverter's or owner's wallet, offering a more privacy-preserving solution. Inverter owners should have the capability to request VCs from the manufacturer on behalf of the inverters, with the owner's identity potentially being linked via the same blockchain or other mechanisms. Additionally, the security of the firmware update process can be further enhanced by integrating antivirus scans for downloaded firmware.

The VPP operator possesses the flexibility to implement more granular levels of the trust cycle, contingent on their interactions. Furthermore, operators can consider their ongoing interactions with inverters when determining the necessary trust state for their processes. Additionally, more detailed actions can be devised to restrict the extent of interactions with inverters based on their trust state.

3 Use case 2: Security Metadata for Secure VPP Operations

3.1 Introduction

The Technology and Cyber Security Assessment of the EDGE project [4], conducted by EY [6], has highlighted a significant concern over the experimented Aggregator setup where entities such as DNSPs, market and system operators, OEMs, or aggregators lack a method to verify if an inverter complies with the expected configurations. If an inverter is already compromised, adversaries can easily mimic the capabilities and configuration of the inverter as needed. This situation can mislead stakeholders into believing that the inverter is functioning as expected or adhering to the anticipated settings, when in fact it may be doing the opposite. This scenario could lead to catastrophic outcomes for the network and related equipment. Therefore, the presence of a reliable mechanism to ensure the conformity of inverter capabilities and configurations is crucial for the proper operation of connected inverters.

3.2 Verify Inverter Capabilities

There is a wide range of capabilities and limitations among inverters, and accurately identifying these capabilities is essential in smart grid operations to ensure optimal calculations for various processes. Therefore, a reliable mechanism for obtaining inverter capabilities is crucial. The simplest approach is to implement an API at the manufacturer's end, allowing interested stakeholders to query the API based on the inverter model. However, this method requires stakeholders to directly communicate with the manufacturer, which may not always be practical, especially considering the potential lifespan of an inverter and the possibility that the manufacturer may no longer be accessible.

To address these shortcomings and provide a more robust solution, we propose that inverter manufacturers establish a ``root of trust" regarding the capabilities and limits of the inverters they produce. However, our assumption is that stakeholders should not need to rely solely on the manufacturer to obtain inverter capabilities once the inverters are installed on customer premises. Instead, stakeholders can rely on manufacturer-issued VCs that certify the inverter's capabilities. These VCs are held by the inverters or their owners, providing a more accessible and decentralised means of verifying inverter capabilities. Alternatively, capability VCs can be issued per each different inverter that the manufacturer produces and store them on a verifiable data registry such as a blockchain. This approach reduces the overhead of issuing a capability VC for every inverter. Interested stakeholders can refer to the VCs on verifiable data registry to obtain capabilities of a specific inverter based on the unique information such as inverter model, batch, and manufactured year.

Listing 6 presents the VC that we have designed to convey the capabilities of an inverter, based on data obtained from Selectronic. We have categorised these capabilities into six distinct groups: power, energy, voltage, current, temperature, and other. The Listing includes several potential attributes within these categories. However, it is important to note that due to the diverse range of capabilities among inverters, manufacturers can make decisions regarding which specific attributes to include, providing additional information to relevant stakeholders. Nevertheless, having a predefined and commonly agreed-upon set of attributes would greatly assist in standardisation efforts.

```
{
 ″@context″:[..],
 ″id″: ″did:abc:sg:inverter:vc:123456789″,
 ″type″: [″VerifiableCredential″, ″InverterIdentityVC″],
 ″issuer″: ″did:abc:sg:manufacturer:123456789″,
 ″validFrom″: ″2023-04-01T10:11:12Z″,
 ″credentialSubject″: {
   ″inverter″: {
     ″manufacturer″: <manufacturer DID>,
     ″modelNo″: <inverter model no>,
     ″year″: <manufactured year>,
     ″batch″: <batch no>
   },
   ″capabilities″: {
     ″power″: { .. },
     ″energy″: { .. },
     ″voltage″: {
       ″voltageMin″: ″″,
       ″voltageMax″: ″″
     },
     ″current″: { .. },
     ″temperature″: {
       ″transformerTempMax″: ″″,
       ″heatsinkTempeMax″: ″″,
       ″batteryTemperatureMax″: ″″,
       ″internalTempMax″: ″″,
       ″powerModuleTempMax″: ″″
     },
     ″other″: { .. }
   }
```

Listing 6: Sample VC for an inverter capabilities

3.3 Inverter Configuration Compliance Check

There is a notable absence of a standardised mechanism for verifying the configuration of an inverter, as highlighted in the assessment report of Project EDGE. This gap creates a security vulnerability where adversaries and unscrupulous users can maintain configurations that are not in line with common operational standards or exploit them for financial advantages, disregarding the common benefits. To address this issue, we have employed VCs and RA as a means of establishing a dependable method for ensuring the integrity of inverter configurations.

Figure 7 depicts the abstracted steps and the entities involved associated with the configuration compliance check process. We propose to have an updated VC for the current inverter configuration. Therefore, inverter/owner require to request a new VC for its configurations upon any changes to settings. This request should contain the owner DID, inverter DID, inverter's current VC, and nonce. The message should be digitally signed by the inverter/owner private key and encrypted using the manufacturer's public key. The manufacturer can verify the digital signature using the keys stated on the DID document of the inverter / owner. Current VC of the inverter provides the ownership affirmation.

Upon validation of inverter ownership, the manufacturer constructs the RA challenge for the inverter and sends the constructed RA challenge to the inverter. Without, restricting to a specific RA implementation, we assume the existence of at least a single predefined RA method that supported by the inverter. Additionally, we emphasise the successful bootstrapping, as it initiates the root-of-trust for the RA process. The inverter generates the response to the received RA challenge and sends it to the manufacturer. The manufacturer verifies the attestation response depending on the utilised RA method. A successful validation leads the manufacturer to issue a new VC for the inverter, confirming its current configurations. At the same time, the manufacturer invalidates previously issued VC for the inverter as it is invalid with the updated configurations. The inverter/owner securely stores the received VC and shares it with the relevant stakeholders such as DNSP and VPP operators, upon request to state their current configurations.

DNSPs/VPP operators can confirm the authenticity of the presented VC by utilising the manufacturer's key as stipulated in the manufacturer's DID document. However, even with the VC's authenticity assured, RA-based compliance configuration checks remain susceptible to TOCTOU (time of check, time of use) attacks. More specifically, we consider the adversaries (not dishonest users) who are manipulating the configurations. To address this vulnerability, we advocate the integration of capability VCs and firmware tracking VCs, thereby introducing an additional layer of trust into the compliance assurance process.

The compliance assurance process at the DNSP/VPP operator is illustrated in Figure 8. In contrast to the commonly adopted compliance states of '*compliance*' and '*not-compliance*', we have introduced an additional state named `semi-compliance' in our model. An inverter is categorised as `not-compliance' if the RA process fails. If the RA is successful and the inverter is found to be in the expected configuration, further checks are conducted against the inverter's capabilities (capability VC) to ensure that the attested configurations align with plausible settings. This step is essential as adversaries may compromise the inverter but manage to evade configuration alterations. Any discrepancies between the attested configurations and the manufacturer-defined capabilities result in a classification of `not-compliance.' The predetermined trust status of firmware updates is taken into consideration on successful validation attested configurations against capabilities.

Inverters running compromised firmware versions, firmware with known vulnerabilities, or those not updated in a timely manner are at risk of compromise. In cases where an inverter is compromised, there is a possibility that it attempts to deceive the RA process by falsely claiming to possess the expected configurations. Therefore, even if an inverter's configurations pass the RA and align with the capabilities, we propose categorising the inverter as `not-compliance' if it falls under a '*Distrust*' state based on its firmware update process. Inverters are classified as `Semi-compliance' if they fall under a '*Semi-trust*' state (according to the firmware update process). Only inverters that are in a `trustable' state, successfully pass RA, and align with capabilities are considered compliant.

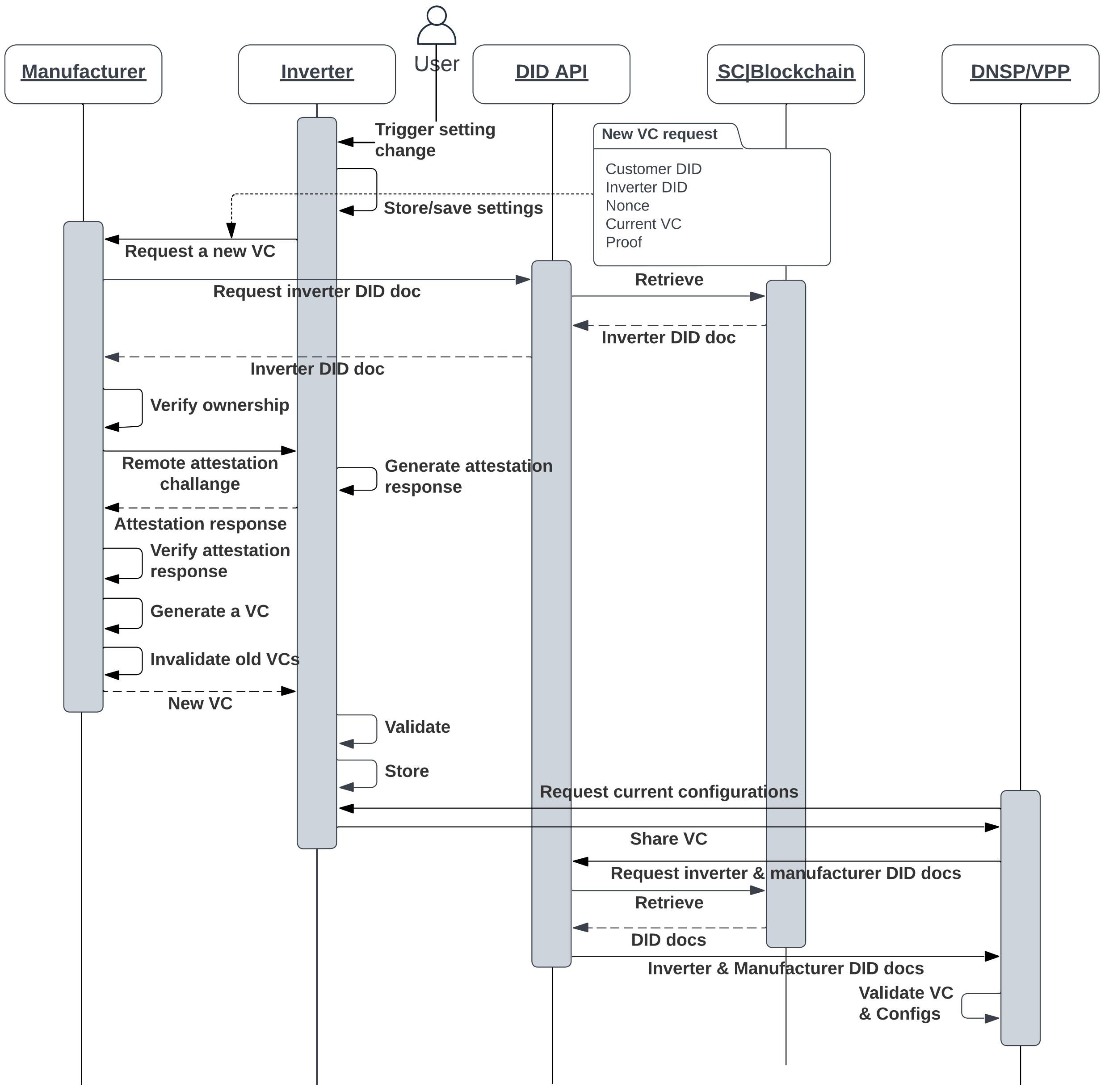

Figure 7: Sequence diagram for the RA based configuration compliance check

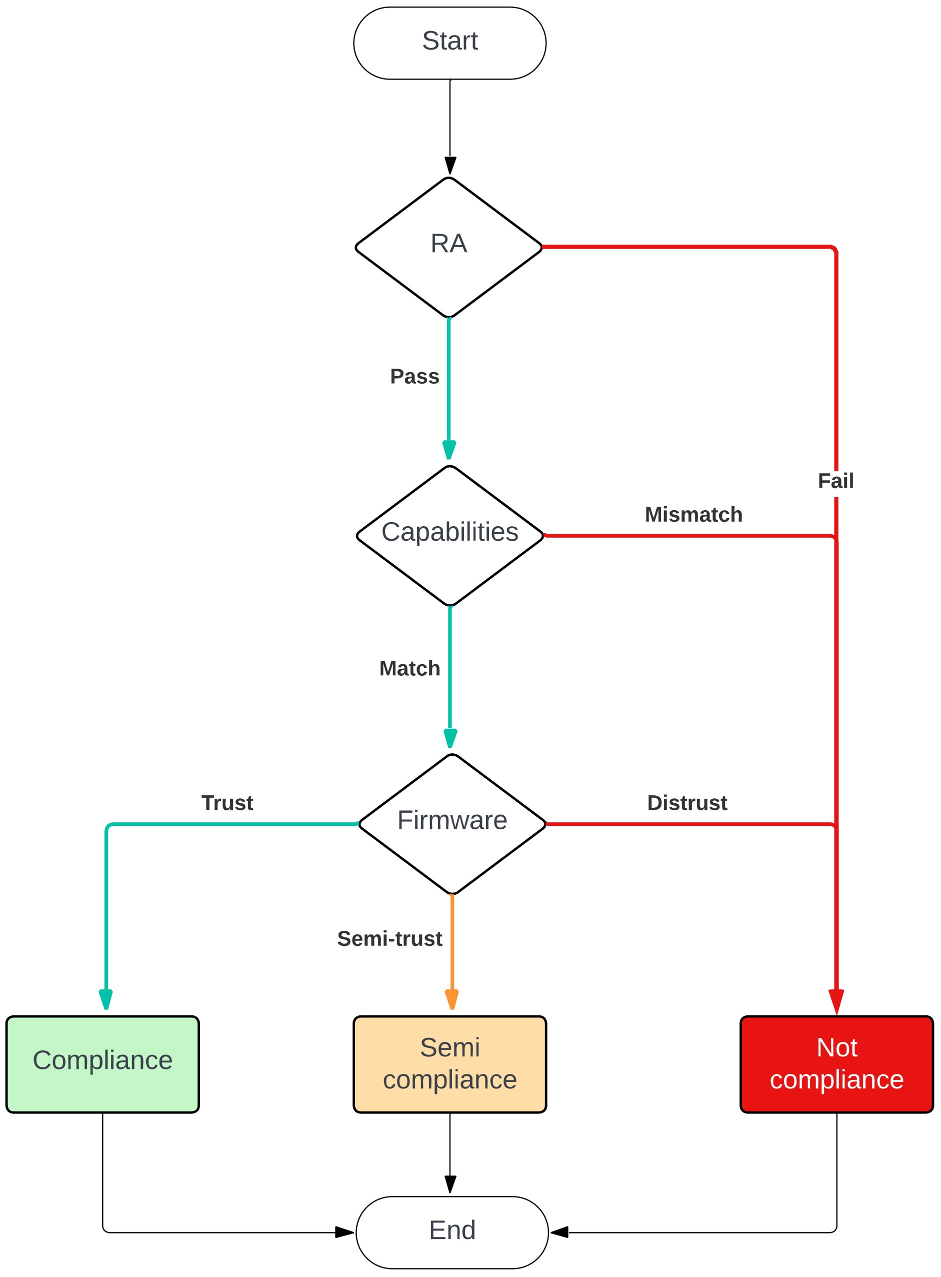

Figure 8: Flow chart for compliance assessment

3.4 Discussion

Despite the practical convenience and the higher level of trust associated with depending on manufacturers for RA of their inverters, there is a potential concern if a particular manufacturer's operations cease or if they are unable to provide RA services for any reason. In such cases, there is a need for a robust and reliable redundancy mechanism for RA verification. This redundancy ensures that RA can still be effectively carried out, even in the absence of the manufacturer. These backup verifiers must possess the necessary tools, knowledge, and infrastructure to conduct RA, including the ability to verify the authenticity and integrity of the inverter's firmware and configurations. This approach enhances the overall resilience and reliability of the proposed process.

4 Discussion

Large-scale implementation of advanced smart grid concepts, including VPPs, aggregators, and energy trading platforms, will be realised in the near future, given the rapid increase in the use of DERs at the consumer level. Hence, the use of smart inverters in provisioning DERs is inevitable. Thus, the security of smart inverters must be thoroughly analysed and evaluated to ensure the reliability and safety of critical infrastructure operations, as the hazardous impact can cause loss of lives and significant financial damage.

Figure 9 shows the abstracted architecture of the proposed system. The manufacturers can either implement their own DID implementation or leveraged an existing DID method. Depending on the utilised DID approach the DID documents can be stored on a blockchain, distributed file system such as IPFS or in a central database. This flexibility allows the manufacturers to decide the most suitable implementation considering the privacy of data and other involved entities. Inverters may initiate direct communication with the DID API or the manufacturer depending on the smartness of the inverter. However, inverter owner would initiate the communication with these entities in most scenarios.

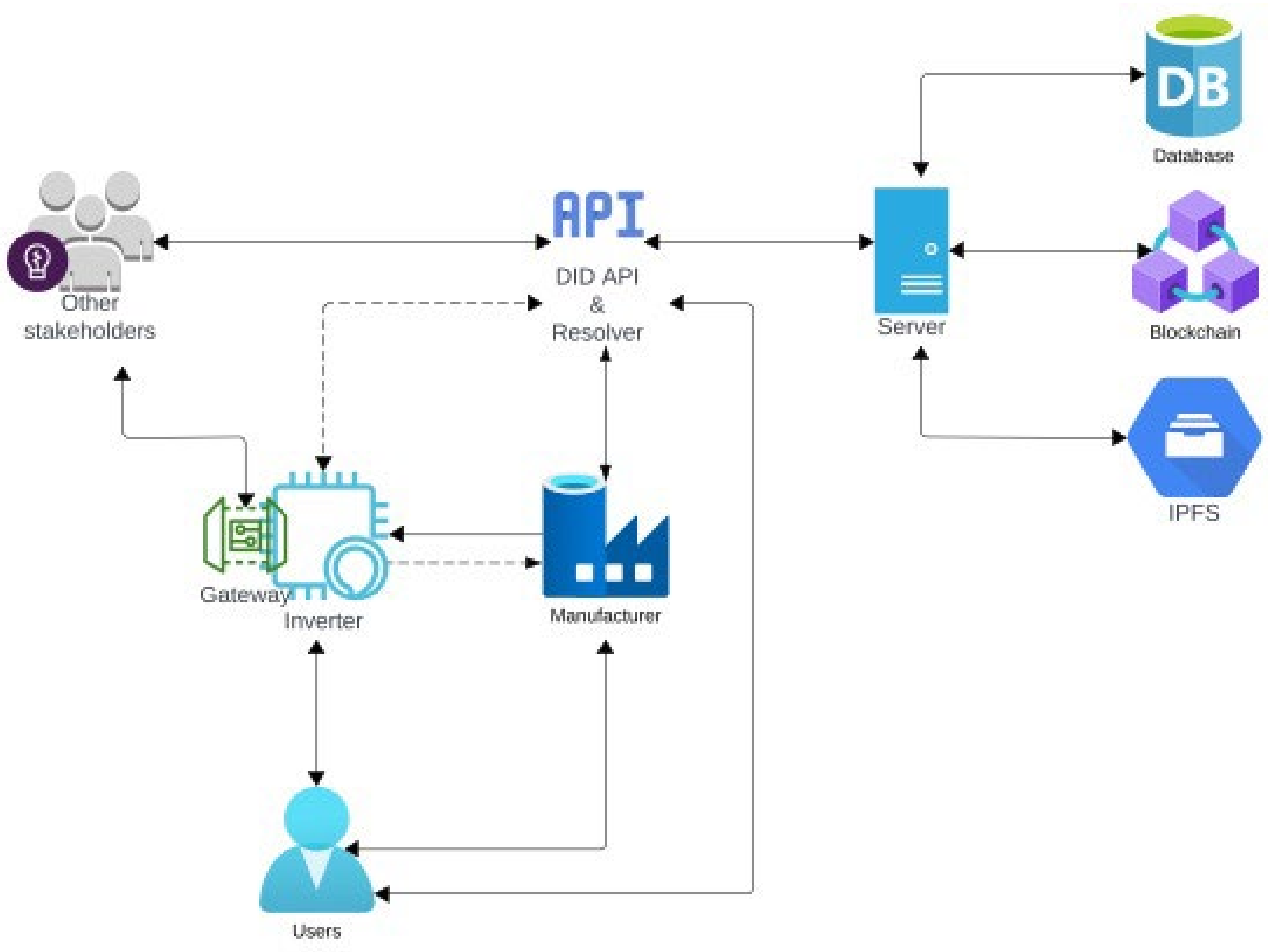

Figure 9: Abstract architecture of the proposed system

4.1 Comparison with IEEE 2030.5 and CSIP

IEEE 2030.5 and the Common Smart Inverter Profile (CSIP) are widely used and commonly adopted application protocols/standards and implementation guidelines in the smart inverter context worldwide. Table 2 compares the existing standards with the proposed solution. It should be noted that while existing standards cover the entire inverter life cycle, our solution focuses solely on analysing the firmware update process. However, we have considered the potential to extend our proposed solution in this comparison.

CSIP and IEEE 2030.5 standards utilise the fingerprint of manufacturer-issued device certificates to derive inverter identities (Long-form Device Identifier and Short-form Device Identifier). This methodology provides a static identity for inverters under the assumption that device certificates are valid indefinitely according to the specification. However, real-world scenarios often require the revocation or renewal of initial issued certificates and keys. Thus, maintaining the longevity of inverter identity becomes complex, as it is closely tied to the device certificate. In contrast, our approach assumes that a manufacturer assign DID, eliminating the tight coupling with a specific key. This allows for key pair revocation and renewal without altering the identity. Furthermore, the use of a one-way hash chain can address this limitation if the DID is generated based on a private/public key [7].

Leveraging DIDs makes key management (rotation, revocation, renewal) standardised and convenient, whereas the existing standards do not specify the key management process. DIDs utilise distributed trust in contrast to the traditional centralised PKI-based root-of-trust approach. This distributed trust enables verification for inverters, even if the manufacturer's existence is compromised in the future, considering the long lifespan of inverters.

The IEEE 2030.5 standard specifies 12 device attributes, including location, identity, firmware version, and hardware version, under *'DeviceInformation'*. These attributes help in constructing a more reliable and precise assessment of inverters, given that the utilities and VPP operators do not have complete control or visibility over inverters. Our design extends this further by allowing manufacturers to utilise VCs in sharing additional information with appropriate security mechanisms, ensuring that authorised entities can access only the necessary information. Similarly, manufacturers can utilise the VC based approach to specify additional capabilities, functionalities, and limits of their inverters in addition to that are already covered under the IEEE standard. Further, regulators can define VC schemes to standardise the additional information and capabilities instead of duplicating schemes.

Despite the current availability of attributes and capabilities, existing standards and guidelines lack a mechanism to track changes to these attributes and capabilities over time. Additionally, existing standards does not focus on device trust or configuration compliance. Furthermore, adversaries can easily fabricate inaccurate information, such as location, capabilities, and firmware version, especially after a firmware/software compromise as the standards do not encompass any post-compromise aspects. Although the proposed model does not eliminate attacks, it offers a mechanism for stakeholders to assess the risk associated with compromised inverters. This enhanced assessment capability empowers stakeholders to adjust their interactions with these inverters. Moreover, our approach has the potential for extension to incorporate the historical changes in attributes and capabilities. Much like the history of firmware updates, having access to this information can facilitate comprehensive anomaly detection and risk assessment for operators.

Table 2: Comparison with IEEE 2030.5/CSIP

Criteria	IEEE 2030.5/CSIP	Proposed Model
Identity	Derived from the certificate fingerprint	DID
Validity	Indefinitely	Customisable
Certificate Revocation	Not specified	Supported via DID docs
Root of Trust	PKI	Distributed
Device information	Limited	Unlimited
Device capabilities	Supported	Supported
History of attributes	Not supported	Supported via VC
Format for device information	Not specified	VC
Format for capabilities	Not specified	VC

4.2 Common VC for Inverters

Smart inverters serve various use cases within the context of the smart grid. However, shared functionalities exist in many of these scenarios. In this section, we classify these shared functionalities and formulate VCs scheme for each one. The categorisation and VC structures can be leveraged in future standardisation initiatives and eliminate the necessity for multiple schemes addressing identical functionalities.

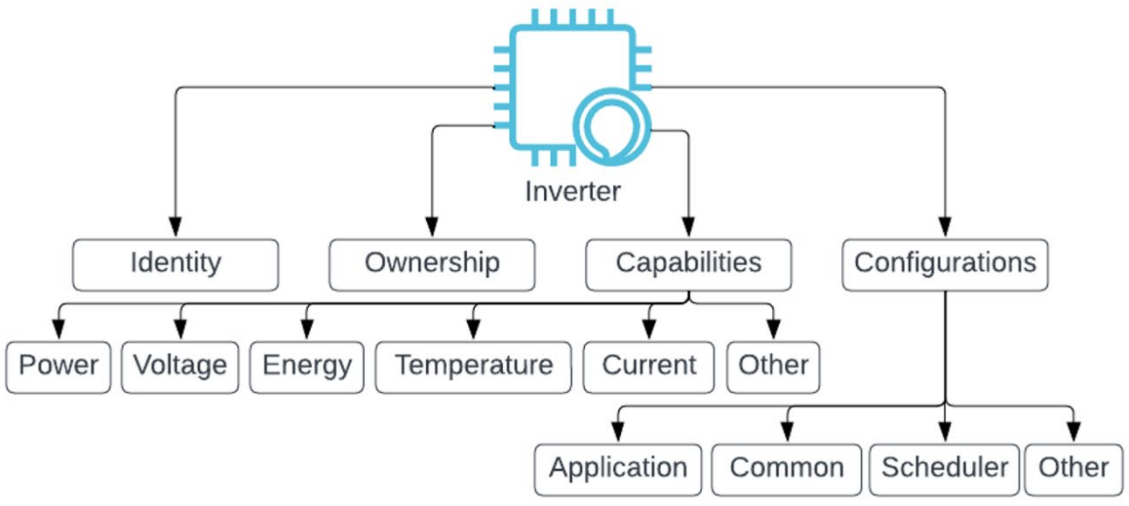

Figure 10: High-level breakdown of inverter functionalities

4.2.1 Identity

The identity of an inverter is an essential part of any process that involves inverters. In particular, the identity stands as a paramount factor in the context of security as it helps in establishing the trust on the device and its data. A secure and reliable identity helps to enforce effective access controls and appropriate monitoring, which mitigate numerous security concerns.

We have used commonly available attributes of inverters to define the Identity VC, which establishes the device identity as depicted in Listing 7. This VC essentially encapsulates the device's DID along with a collection of static attributes, encompassing device characteristics and manufacturer details. These attributes remain consistent throughout the device's operational lifespan. Furthermore, we recommend integrating certain device metadata, such as the current firmware version and the timestamp of the last update, into the identity VC to provide additional insights on device status.

The identity VC must only be issued by the corresponding manufacturer. The initial identity VC is issued and digitally signed by the manufacturer before the inverter leaves the factory. Depending on the inverter's storage capacity, this identity VC can be embedded within the inverter or stored in a secure location. If the inverter lacks the capacity to store its VCs, the VC must be securely transferred to the end user through the inverter supply chain.

4.2.2 Ownership

Many smart grid use cases involve in requiring the ownership of inverters, which associates the inverter to its owner/DER given the distributed smart grid setups and diverse range of inverters/DERs. Accurate representation of the ownership can prevent adversaries from misusing the inverter. Additionally, appropriate access control mechanisms can be implemented to restrict changes to configurations given the availability of accurate ownership details.

```
{
  "@context":[..],
  "id": "did:abc:sg:inverter:vc:123456789",
  "type": ["VerifiableCredential", "InverterIdentityVC"],
  "issuer": "did:abc:sg:manufacturer:123456789",
  "validFrom": "2023-04-01T10:11:12Z",
  "credentialSubject": {
    "immutable": {
        "id": "did:abc:sg:inverter:123456789",
        "serialNo": "123456789",
        "manufacturedDate":"2021-01-01T00:01:02Z",
        "manufacturerName": "ABC",
        "modelNumber": "XYZ",
        "batchNumber": "12345"
    },
    "dynamic": {
        "firmwareVersion": "1.2.3",
        "lastUpdatedDate": "2023-04-01T00:01:02Z"
    }
  },
  "credentialSchema": {..},
  ...
  "proof": {..}
}
```

Listing 7: Sample VC for an inverter identity

The proposed ownership VC is shown in Listing 8. It consists of the inverter DID, the owner DID, and the date of purchase.

```
{
  "@context":[..],
  "id": "did:abc:sg:inverter:vc:123456789",
  "type": ["VerifiableCredential", "InverterIdentityVC"],
  "issuer": "did:abc:sg:manufacturer:123456789",
  " validFrom": "2023-04-01T10:11:12Z",
  "credentialSubject": {
    "owner": "did:abc:sg:user:123456789",
    "inverterId": "did:abc:sg:inverter:123456789",
    "purchasedDate" : "2022-01-01T00:01:02Z"
  },
  "credentialSchema": {..},
  ...
  "proof": {..}
}
```

Listing 8: Sample VC for an inverter ownership

4.2.3 Capabilities

Inverters possess diverse range of capabilities, with some of these capabilities holding significant importance for external stakeholders, while others are more for internal use. To effectively manage this diversity, we propose categorising these capabilities into six high-level groups, as illustrated in Listing 9. The manufacturer has the flexibility to decide which specific attributes should be included within each category. For instance, the VC highlights a fundamental capability related to voltage regulation, where the inverter can operate within a broad voltage range. This capability assumes an essential role in sustaining grid stability by ensuring that the electrical output aligns the grid's voltage specifications without triggering voltage instability, especially with high penetration of solar PVs.

The "other" category specified in the VC offers the flexibility to accommodate any additional functionalities. Inverters often extend their support to advanced features, including grid-tied operation, reactive power control, voltage regulation, and the facilitation of communication protocols.

We expect manufacturers to issue capability VC before the inverter leaves the manufacturing facility. Similar to identity VC, this VC can be stored on the inverter or shared to the user over the supply-chain.

```
{
 ″@context″:[..],
 ″id″: ″did:abc:sg:inverter:vc:123456789″,
 ″type″: [″VerifiableCredential″, ″InverterIdentityVC″],
 ″issuer″: ″did:abc:sg:manufacturer:123456789″,
 ″validFrom″: ″2023-04-01T10:11:12Z″,
 ″credentialSubject″: {
  "inverterId": <inverter DID>,
  ″power″: { .. },
  ″energy″: { .. },
  ″voltage″: {
     ″voltageMin″: ″5V″,
     ″voltageMax″: ″230V″
  },
  ″current″: { .. },
  ″temperature″: { .. },
  ″other″: { .. }
 },
 ″credentialSchema″: {..},
 ...
 ″proof″: {..}
```

Listing 9: Sample VC for an inverter capabilities

4.2.4 Configuration

Inverter configuration, as shown in the Listing 10, plays a primary role in the seamless integration of DERs into smart grid operations including VPP. These configurations encompass a wide array of parameters, ranging from application-specific settings to battery management and communication protocols. Alteration to inverter configurations can impact the performance, efficiency, and reliability of DERs within the grid.

One critical aspect of inverter configuration is specifying the operating parameters for grid interaction.

```
{
  ″@context″:[..],
  ″id″: ″did:abc:sg:inverter:vc:123456789″,
  ″type″: [″VerifiableCredential″, ″InverterIdentityVC″],
  ″issuer″: ″did:abc:sg:user:123456789″,
  ″validFrom″: ″2023-04-01T10:11:12Z″,
  ″credentialSubject″: {
    ″inverterId″: <inverter DID>,
    ″common″: { .. },
   ″application″: { .. },
   ″battery″: { .. },
   ″scheduler″: { .. }
  },
  ″credentialSchema″: {..},
  ...
  ″proof″: {..}
```

Listing 10: Sample VC for an inverter configurations

This includes defining acceptable voltage and frequency ranges, both on the primary and alternate power sources, to ensure that the inverter operates within the grid's stability constraints. Battery-related configurations, like the maximum charge capability and compensation temperature coefficients, are instrumental in managing energy storage systems effectively. By setting appropriate limits and thresholds, the inverter can optimise the charging and discharging processes, extending the lifespan of the battery, and ensuring that it operates within safe operating conditions. This is particularly vital for grid resilience during peak demand periods, where stored energy can be strategically discharged to alleviate stress on the grid.

As explained in earlier sections, we expect manufacturers to issue configuration VCs for their inverters verifying the current configuration (via RA or other mechanism) upon user request.

4.3 Protocols for VC Issuance and Exchange

4.3.1 VC Issuance

The issuance of VCs is a critical component within the proposed architecture. This section outlines the protocol designed for issuing VCs to inverters. It's important to note that we abstract the claim verification process that occurs before issuance and assume the existence of a reliable mechanism for the manufacturer to validate the presented claims. Algorithm 3 provides an overview of this proposed protocol.

The process is initiated by a user or inverter, who requests the DID document of the corresponding manufacturer from the DID API, utilising the embedded manufacturer DID. Extracting the manufacturer's public key from this document is essential for encrypting messages sent to the manufacturer. To construct the message, various data components are concatenated, including the existing VC of the inverter, the DIDs of the user and inverter, other relevant data and nonce. The user or inverter employs their private key to sign this constructed message. The resultant message and signature are combined to form the new VC request. The manufacturer's public key, extracted earlier, is then used to encrypt this request, which is subsequently transmitted to the manufacturer's designated endpoint, either declared in the DID document or via another means.

Upon receiving the request, the manufacturer decrypts it using its private key and extracts the DIDs from the request. The corresponding DID documents for these DIDs are retrieved with the assistance of the DID API. The public keys of the requested inverter and user are extracted from their respective DID documents, which are then used to verify the authenticity and integrity of the received request. Furthermore, the embedded inverter's VC is validated using the manufacturer's public key. To ensure it has not been revoked or expired, the validity of the VC is cross-referenced with the credential status list.

Following the successful validation of the received request, the manufacturer proceeds to verify the validity of the claims using existing mechanisms. A new VC is generated to reflect the verified claims, leveraging an existing scheme for reusability and wider acceptance. This new VC is signed with the manufacturer's private key. To prevent future misuse, the manufacturer revokes or invalidates any older VCs previously issued for the inverter upon generating the new VC.

The manufacturer's DID and the new VC, signed with the manufacturer's private key, are concatenated to form a message. This message is then encrypted (combination of the signature and message), using the public key of the requested inverter or user, which was extracted from their corresponding DID documents. The encrypted message is transmitted back to the inverter or user, who uses their private key to decrypt the response. The authenticity of the message is verified using the manufacturer's public key. The received VC is subsequently stored within a digital wallet, provided its signature is successfully validated.

Algorithm 3: VC Issuance

```
$U/SI \rightarrow DIDAPI$
    Request corresponding DIDDocs
        $DIDDoc^{M_i} \leftarrow didDoc(DID^{P}_{M_i})$
$U/SI$
    Extract key from DIDDocs
        $PK^{DID}_{M_i} \leftarrow verificationKey(DIDDoc^{M_i})$

    Construct a new VC request
        $msg = VC^{SI_iU_i}_{M_i} || DID^{P}_{U_i} || DID^{P}_{SI_i} || data || nonce$
        $sig = sign(hash(msg), SK^{DID}_{U_i/SI_i})$
        $vcReq = msg || sig$

    Encrypt the message using Manufacturer's public key
        $encryptedVcReq = encrypt(vcReq, PK^{DID}_{M_i})$
$U/SI \rightarrow M$
    Send the encrypted new VC
$M$
    Decrypt the request
        $decryptedVcReq = decrypt(encryptedVcReq, SK^{DID}_{M_i})$

    Extract the DIDs from the message
        $DID^{P}_{U_i}, DID^{P}_{SI_i} \leftarrow extractDidsFromMsg(decryptedVcReq)$
$M \rightarrow DIDAPI$
    Request corresponding DIDDocs
        $DIDDoc^{U_i} \leftarrow didDoc(DID^{P}_{U_i})$
        $DIDDoc^{SI_i} \leftarrow didDoc(DID^{P}_{SI_i})$
$M$
    Extract keys from DIDDocs
        $PK^{DID}_{U_i} \leftarrow verificationKey(DIDDoc^{U_i})$
        $PK^{DID}_{SI_i} \leftarrow verificationKey(DIDDoc^{SI_i})$

    Verify the signature and request using retrieved keys
        $verify(sig, hash(msg), PK^{DID}_{U/SI_i})$
        $verify(signature(VC^{SI_iU_i}_{M_i}), hash(VC^{SI_iU_i}_{M_i}), PK^{DID}_{M_i})$

    Verify the validity of the VC on the CredentialStatusList

    Verify claims //this is outside of the scope

    Generate a new VC for the claims ($VC^{SI_iU_i}_{M_i}$) and sign
        $sign(VC^{SI_iU_i}_{M_i}, SK^{DID}_{M_i})$

    Invalidate previous VCs
        Set old VCs as revoked/expired on the CredentialStatusList

    Construct the new VC response ($VC^{SI_iU_i}_{M_i}$)
        $msg = VC^{SI_iU_i}_{M_i} || DID^{P}_{M_i} || nonce$
        $sig = sign(hash(msg), SK^{DID}_{M_i})$
        $vcRes = (msg || sig)$

    Encrypt the message using user/inverter's public key
        $encryptedVcRes = encrypt(vcReq, PK^{DID}_{U_i/SI_i})$
$M \rightarrow U/SI$
    Send the encrypted new VC ($VC^{SI_iU_i}_{M_i}$)
$U/SI$
    Decrypt the VC response
        $decryptedVcRes = decrypt(encryptedVcRes, SK^{DID}_{SI_i/U_i})$

    Verify the message signature and VC using manufacturer key
        $verify(sig, hash(msg), PK^{DID}_{M_i})$
        $verify(signature(VC^{SI_iU_i}_{M_i}), hash(VC^{SI_iU_i}_{M_i}), PK^{DID}_{M_i})$

    Store it in the digital wallet
```

4.3.2 VC Exchange

Much like the issuance of VCs, the exchange of VCs constitutes a pivotal aspect of the proposed system. This section explains the protocol for VC exchange, as illustrated in Algorithm 4. We employ the use case of enrolling an inverter in a VPP program to illustrate this protocol.

The process commences with either the inverter or the user obtaining the DID document of the corresponding VPP operator from the DID API, using the operator's advertised DID. From this document, the relevant public keys and associated endpoints are extracted. The enrolment request message encompasses several elements, including the inverter's VC, the DIDs of both the user and the inverter, any additional requisite data for enrolment, and nonce. The private key of the inverter or user is utilised to digitally sign this constructed message. Subsequently, the message and its signature are encrypted using the VPP operator's public key, which was extracted from the DID document, and transmitted to the operator.

Upon receiving the request, the VPP operator begins a series of steps. Firstly, it decrypts the received request using its corresponding private key. Secondly, it extracts the specified DIDs of the inverter and user from the request. Thirdly, it retrieves the DID of the issuer of the embedded VC (inverter manufacturer). These extracted DIDs then facilitate the retrieval of the corresponding DID documents, along with the associated public keys of the involved entities. The public key of the inverter or user is employed to validate the message signature of the request. Simultaneously, the manufacturer's public key is employed to ascertain the authenticity of the inverter's VC, which is embedded within the request. Lastly, the validity of the VC is verified against the credential status list published by the manufacturer.

Upon the successful verification of the VC, the operator proceeds with the compliance check and the enrolment process. It is important to note that the details of these subsequent steps fall outside the scope of this protocol.

Algorithm 4: VC Exchange

```
$U/SI \rightarrow DIDAPI$
    Request corresponding DIDDocs
        $DIDDoc^{DNSP/VPP} \leftarrow didDoc(DID^{P}_{DNSP/VPP})$

$U/SI$
    Extract key from DIDDocs
        $PK^{DID}_{DNSP/VPP} \leftarrow verificationKey(DIDDoc^{DNSP/VPP})$

    Construct an enrol request
        $msg = VC^{SI_iU_i}_{M_i} || DID^{P}_{U_i} || DID^{P}_{SI_i} || data || nonce$
        $sig = sign(hash(msg), SK^{DID}_{U_i/SI_i})$
        $enrolReq = msg || sig$

    Encrypt the message using DNSP/VPP's public key
        $encryptedEnrolReq = encrypt(enrolReq, PK^{DID}_{DNSP/VPP})$

$U/SI \rightarrow DNSP/VPP$
    Request to enrol
        Send the $encryptedEnrolReq$ to DNSP/VPP

$DNSP/VPP$
    Decrypt the request
        $decryptedEnrolReq = decrypt(encryptedEnrolReq, SK^{DID}_{DNSP/VPP})$

    Extract the DIDs from the message
        $DID^{P}_{U_i}, DID^{P}_{SI_i} \leftarrow extractDidsFromMsg(decryptedEnrolReq)$
        $DID^{P}_{M_i} \leftarrow extractDidFromVC(VC^{SI_iU_i}_{M_i})$

$DNSP/VPP \rightarrow DIDAPI$
    Request corresponding DIDDocs
        $DIDDoc^{U_i} \leftarrow didDoc(DID^{P}_{U_i})$
        $DIDDoc^{SI_i} \leftarrow didDoc(DID^{P}_{SI_i})$
        $DIDDoc^{M_i} \leftarrow didDoc(DID^{P}_{M_i})$

$DNSP/VPP$
    Extract keys from DIDDocs
        $PK^{DID}_{U_i} \leftarrow verificationKey(DIDDoc^{U_i})$
        $PK^{DID}_{SI_i} \leftarrow verificationKey(DIDDoc^{SI_i})$
        $PK^{DID}_{M_i} \leftarrow verificationKey(DIDDoc^{M_i})$

    Verify the signature and request using the retrieved keys
        $verify(sig, hash(msg), PK^{DID}_{U/SI_i})$
        $verify(signature(VC^{SI_iU_i}_{M_i}), hash(VC^{SI_iU_i}_{M_i}), PK^{DID}_{M_i})$

    Verify the validity of the VC on the CredentialStatusList

    Check compliance//this is outside of the scope
    Enrolment //this is outside of the scope
```

5 Conclusion & Next Steps

Large scale VPPs and other setups that utilise consumer owned DERs are imminent with the high penetration of DERs and regulations towards greener energy. Despite the advantages of these setups, they enlarge the attack surface for adversaries, as regulators and other entities do not have complete control or visibility over the consumer-owned resources.

In the first phase project, we have investigated methods to improve the security of smart inverter integration into smart grid operations. In particular, we considered the security of VPP operations that utilise consumer owned inverters leveraging device identification, configuration management, device trust, and data integrity with the use of decentralised identity and blockchain technology. More specifically, we made the following contribution from this project.

1. **Enhanced Firmware Security:** We introduced a novel approach based on VCs to ensure the security of the firmware update process. By leveraging VCs, we provide a secure and tamper-evident method to track the history of firmware updates, enhancing the overall integrity and trustworthiness of smart inverters.
2. **Trust-Cycle for Inverters:** To establish secure interactions between inverters and VPP operators, we proposed a trust-cycle model. This model is based on the historical record of firmware updates, enabling VPP operators to assess the trustworthiness of each inverter's configuration dynamically.
3. **Capability Verification:** We introduced a VC-based method for verifying inverter capabilities. This ensures that the reported capabilities of each inverter align with their actual functionalities, reducing the risk of malicious or misconfigured devices compromising grid operations.
4. **Configuration Compliance:** To address the challenge of ensuring inverter configuration compliance, we integrated inverter capabilities with trust states. This approach helps stakeholders verify that inverters' configurations match their specified capabilities, enhancing overall system reliability.
5. **Standard Enhancements:** We identified potential vulnerabilities and issues within the existing IEEE 2030.5-2018 standard, particularly in the context of cybersecurity. To address these concerns, we proposed architectural enhancements that can enhance the security and resilience of smart grids.
6. **Common Use Case VCs:** Recognising the versatility of VCs, we explored the potential of defining VCs for common use cases in the realm of smart inverters. This initiative aims to standardise and simplify trust-related processes across various scenarios.

This work significantly contributes to the security, trust, and reliability of smart inverters and DERs within the evolving landscape of modern energy grids. By harnessing the power of VC and innovative trust models, we pave the way for safer, more resilient, and more efficient energy systems. These advancements have the potential to shape the future of smart grid technology, ensuring its readiness for the challenges and opportunities that lie ahead.

6 Reference

[1] Solar energy. https://arena.gov.au/renewable-energy/solar/. Accessed: 2023-04-14.

[2] Hedayat Saboori, M Mohammadi, and R Taghe. Virtual power plant (vpp), definition, concept, components and types. In 2011 Asia-Pacific power and energy engineering conference, pages 1–4. IEEE, 2011.

[3] Manu Sporny, Dave Longley, Markus Sabadello, Drummond Reed, Orie Steele, and Christopher Allen. Decentralized identifiers (dids) v1.0. Technical report, W3C, 2021.

[4] Project edge. https://aemo.com.au/en/initiatives/major-programs/nem-distributed-energy-resources-der-program/der-demonstrations/project-edge. Accessed: 2023-09-04.

[5] Manu Sporny, Dave Longley, and David Chadwick. Verifiable Credentials Data Model 1.1. Technical report, W3C, 2022.

[6] Project edge technology and cybersecurity assessment. Technical report, 2023.

[7] Chang-Seop Park and Hye-Min Nam. A new approach to constructing decentralized identifier for secure and flexible key rotation. IEEE Internet of Things Journal, 9(13):10610–10624, 2022.

www.racefor2030.com.au

RACE for 2030

Australian Government
Department of Industry, Science and Resources

AusIndustry
Cooperative Research Centres Program