Table of Contents

Chapter 1

Introduction

The current state of digital identity management has matured sufficiently to cater to the growing demand for secure and user-centric authentication in an increasingly digital world [11]. Previously, identity systems were centralized, whereby each organization required the authentication of an individual using different credentials which led to problems such as password fatigue, scalability issues, and increased vulnerability to data breach. Federated Identity Management (FIM) gave rise to Single Sign-On (SSO) mechanisms, such as in OAuth 2.0, SAML, or OpenID Connect, allowing the access of several services with just one trusted Identity Provider (IdP). Although these reduced the number of authentication hurdles, they still required a central authority, which means possible privacy invasions, data-tracking, and single points of failure [10]. Therefore, the advent of Self-Sovereign Identity (SSI) came to address these problems as a de-centralized approach wherein individuals truly own their digital identities. DIDs (Decentralized Identifiers) and VCs (Verifiable Credentials) are the core principles forming SSI that can make possible very secure and tamper-proof identity verification without reliance on an intermediary through the combination of blockchain technologies [8]. This means that a person would generate a DID for himself, acquire credentials from trusted issuers (like governments or educational institutions), and then be able to store them in digital wallets. When such credentials need to be verified, that presentation can be done to the verifiers who can do that without having to rely on a central authority to achieve privacy through selective disclosure and cryptographic proofs [1]. Such an approach does not just empower individuals with control over their personal data but also reduces risks of having such data in centralized data repositories, such as large-scale breaches. Another advantage of SSI is promoting interoperability across several platforms and services, making much easier experiences for the user while

also creating confidence in digital interactions [20]. However, SSI adoption does not come without hurdles [17]. Backwards compatibility with systems already in existence, individual resistance to adaption, and the entrenched learning curve in decentralized technology are major deterrents towards the adoption of this model. Moreover, standardization will be needed to facilitate succeeding interoperability among diverse sectors among different fragmentation avoided. But with this, the advantages that may be gained through SSI concerning the security of data, privacy of the user, and operation efficiency make this concept more favorable in areas such as healthcare, finance, and public services [3]. The embrace of this approach does a revolutionary job in restructuring digital identity, providing a futuristic pathway for a solution in empowerment of the user and the democratization of trust in a digital age.

1.1 Problem Statement

The ownership of electronic appliances, particularly mobile phones, has presented myriad challenges in the present day. Mobile phones, unlike vehicles that are officially registered and tracked through government databases, lack any universal or standardized ownership registry. The absence of such a traceable legitimacy poses a glaring vulnerability in terms of digital security and accountability [16]. In other words, when a device is lost or stolen, there is no reliable way to prove which person owns it, rendering any attempts to recover the device impossible. Even worse, resetting stolen phones, wiping them clean of data, and trading them off with black-market vendors create a situation that almost makes it impossible for anyone to reclaim their devices. Stolen electronics are pouring onto black markets nowadays because these loopholes allow for untraceable sale. It also gives birth to the very act of theft, making it difficult for anyone to claim ownership of stolen electronics. Consider the situation where someone's phone is stolen but later recovered by the police: there is currently no reliable means of establishing that the phone in question is the rightful property of the original victim. This real failure serves to accentuate a glaring weakness in contemporary device identity management and stands as a dire call for the development of a reliable and secure ownership rights documentation, verification, and protection framework. The absence of such a mechanism promotes criminal activity and destroys consumer confidence, both of which are conspicuous anomalies in the present management practices of digital properties.

1.2 Solution

This study presents an SSI-based solution to the problem of verifying ownership of mobile and electronic devices, which currently lack a centralized registration system. The proposed decentralized architecture binds ownership to users through cryptographic signatures and blockchain storage. Upon purchasing a device, the user signs a combination of their Decentralized Identifier (DID) and the device's unique ID (IMEI, serial number, or UUID) using their private key. The hashed values and signature are stored on the Polygon blockchain, creating a tamper-proof record of ownership. During verification, the system checks for the existence and validity of these records to confirm rightful ownership. The approach ensures data integrity, eliminates the risk of duplicate registrations, and prevents black market resale. By removing the need for central authorities, the system enhances privacy, accountability, and trust in digital transactions.

1.3 Motivation

The rising incidence of mobile device theft, lack of a secure mechanism for ownership verification, and unregulated resale markets now create an extremely urgent demand for a robust device identity management system. Millions of phones are stolen every year; tracking these phones or proving ownership for recovery purposes is next to impossible. Unlike properties where an official ownership transfer registration exists, electronic devices do not have an official registration system, preventing users from verifying their lawful possession while in disputes or theft cases. This gap created all the potentials for black markets whereby stolen or counterfeit devices are readily sold, increasing the cases of fraud and security threats. On top of that, users have no way of asserting any identity over the device, thus creating unauthorized access, data breaches, and privacy loss. The challenge also remains of transferring ownership in a sale using a verifiable and secure mechanism, leading to many disputes regarding trust and misuse. These challenges drive the need for a new decentralized privacy-preserving solution to bind devices to their true owners in a secure way, thus providing trust in digital transactions and reducing black market activities.

Chapter 2

Related Work

2.1 Literature Survey

Digital Identity Management arose as a technological response to the need for user verification and authentication during the early formative phases of the internet. From the very beginning, identity systems were centralized because service providers managed their own databases and authentication schemes in isolation. Users had to register with and manage separate credentials for every platform they used, leading to a series of fragmented identity experiences, widespread password reuse, and compromised security. Interoperability, scalability, and privacy were a big challenge for these systems because service providers had complete control over personal data. The absence of user visibility regarding how their data was stored, processed, or shared raised the alarm bells even louder when digital services started their assault worldwide.

Responding to these limitations, Federated Identity Management came into the scene [19], where the FIM was viewed as a bridge between usability and centralized control with the introduction of trusted third-party Identity Providers (IdPs) who authenticated users under the purview of the multiple service providers freeing the user from remembering many passwords. The model became very popular with the SAML, OAuth, OpenID, and OpenID Connect protocols, enabling Single Sign-On (SSO) [7]—the mechanism that allows the user to authenticate once and to gain access to multiple applications [15]. SSO thus increased user experience and security by centralizing authentication but also created other challenges. The central identity providers became the single point of failure and the entities of mass surveillance, tracking, and profiling user behavior across services. Furthermore, such centralization put the identity control away from the users, which created

serious issues towards privacy, especially in cross-domain scenarios where there are no foundational trust frameworks [10] [6].

At the same time, the Web Services Description Language (WSDL) and Universal Description, Discovery, and Integration (UDDI) stack became widely adopted in service-oriented architectures to describe, locate, and integrate web services [19]. In the world of federated identity, these help automate and discover authentication services. Technically, however, WSDL and UDDI were centralized or semi-centralized models that never offered the capability to decentralize user control over identity or settle any deep-seated privacy issues. These limitations continued even as services grew interconnected and programmable.

The awareness of these deficiencies, among other factors, began to create user consciousness and regulatory pushes, such as in the General Data Protection Regulation (GDPR). From that moment commenced the transition towards decentralized identities, leading to Self-Sovereign Identity (SSI) [18], a concept that rethinks digital identity as an object owned and controlled by the user [11]. In direct opposition to both centralized and federated systems, SSI gives the authority and control of identity information to the user while removing third-party intermediaries from the process of credential presentation and authentication. Based on principles first articulated by Christopher Allen in 2016 and subsequently pushed forward by projects such as the Sovrin Foundation and Hyperledger Indy [3], SSI represents a paradigm underpinned by two interoperability-driven concepts of privacy and autonomy [21].

2.1.1 Core Properties of SSI

- **Existence:** Users must have an independent existence.
- **Control:** Users must control their digital identities without external interference.
- **Access:** Users must have unrestricted access to their own data.
- **Transparency:** Systems and algorithms must be open and understandable.
- **Persistence:** Identities must be long-lived and durable over time.
- **Portability:** Information and credentials must be easily moved across platforms.
- **Interoperability:** Identities must be usable across different systems and networks.

- **Minimal Disclosure:** Only the necessary data should be shared, using techniques such as Zero-Knowledge Proofs (ZKPs).
- **Protection:** Users' rights must be protected, especially with regard to privacy and data use.

SSI is achieved with Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs) [20]. DIDs are user-controlled, unique identifiers anchored on distributed ledgers, providing tamper-resistance and removing the need for centralized registries. VCs, in contrast, are digitally signed, cryptographically secure credentials that can verify different types of claims, e.g., a driver's license or academic certificate. VCs are issued by trusted issuers and possessed by users (holders) in digital wallets. During interactions with service providers (verifiers), users can selectively reveal parts of their credentials using cryptographic methods such as Zero-Knowledge Proofs. Most importantly, personal data is not stored on-chain; only required metadata such as schemas, revocation registries, and DID documents are stored on the blockchain, maintaining privacy but allowing for trust [1].

2.1.2 Ecosystem Services in SSI

- **Issuer:** Trustworthy party that provides verifiable credentials.
- **Holder:** Individual or organization that owns and manages credentials in a digital identity wallet.
- **Verifier:** Entity or agency that validates the authenticity and validity of credentials by cryptographic evidence.

SSI adds to privacy, scalability, and cross-border operability, making it appropriate for verticals like digital government, finance, health, and education. Blockchain immutability and consensus protocols assist in ensuring data integrity without keeping sensitive data on-chain [2] [5]. However, promising as it may be, SSI remains in development and is vulnerable to a myriad of challenges. The field addresses core issues like governance frameworks that properly define the role and responsibility of ecosystem participants as well as interoperability between varied DID approaches and credential types [17]. Key recovery is a high-priority technical challenge, especially when wallet or cryptographic key access is lost by users. High-load credential issuing or verification procedures also must optimize scalability. User experience remains a challenge because SSI wallets must accommodate non-tech users while

adopting advanced features like credential revocation, selective disclosure, and managing expiration [14] [4].

There are several different SSI implementations under development today [8]. Sovrin, an implementation based on Hyperledger Indy, provides privacy-focused verifiable credentials with advanced ZKP capabilities [12]. uPort, an Ethereum implementation, focuses on user-managed identity with blockchain-rooted provenance [13]. Civic provides secure identity verification to assist in ensuring compliance with Know Your Customer (KYC) and Anti-Money Laundering (AML) regulations. Microsoft's ION, based on the Bitcoin blockchain, provides a public, scalable DID implementation. These efforts continue to experiment with various combinations of decentralization, trust models, and cryptographic proof to define the future of digital identity [9].

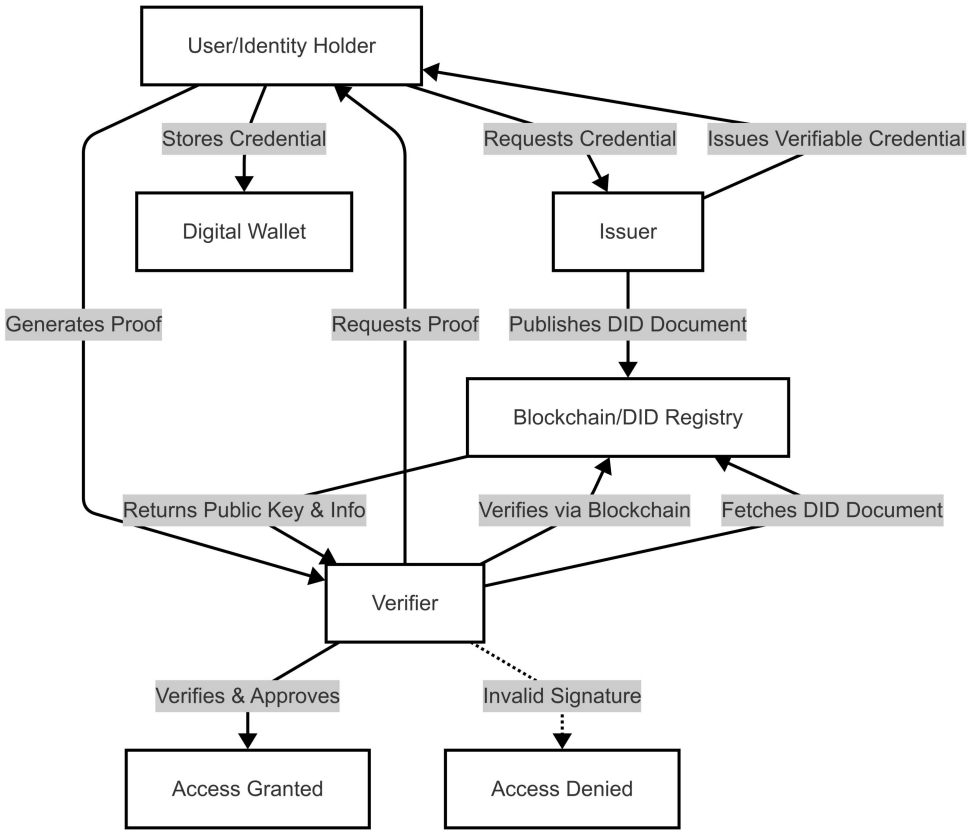

Fig. 2.1 Self-Sovereign Identity Workflow

Chapter 3

Methodology

3.1 Proposed Method

This proposed model sets up a safe and decentralized system to confirm ownership of mobile or electronic devices using self-sovereign identity (SSI) and blockchain technology. Each device can uniquely be identified (for instance, IMEI, serial number, or UUID) in an immutable way. Once bought, the user registers the device by entering their decentralised identifier (DID) through MetaMask and the Device ID. By combining them, a digital signature is created by signing the combination of DID and Device ID with the user's private key and a system private key. This signature, along with the hashed DID and the hashed Device ID, is sent to an immutable location within the Polygon blockchain to build a tamper-proof record. Ownership can, therefore, be confirmed by checking if the stored hashes and signatures correspond to the input, providing assurance that they are not counterfeit. There is a provision to block further attempts to register a device after an initial registration. An important feature of this system is the ability for the users to mark the status of a device on-chain by revoking it if lost or stolen. In turn, if the device is found later, the users can un-revoke it and re-establish their ownership-rights. Re-registration attempts will also send alerts via email and messages automatically to the original owner so that timely action can be taken. This model therefore provides trustless privacy-preserving ownership verification, prevents black market resale, and empowers users with control over their devices.

For a visual representation of workflow, please refer to the sequence diagram fig 3.1:

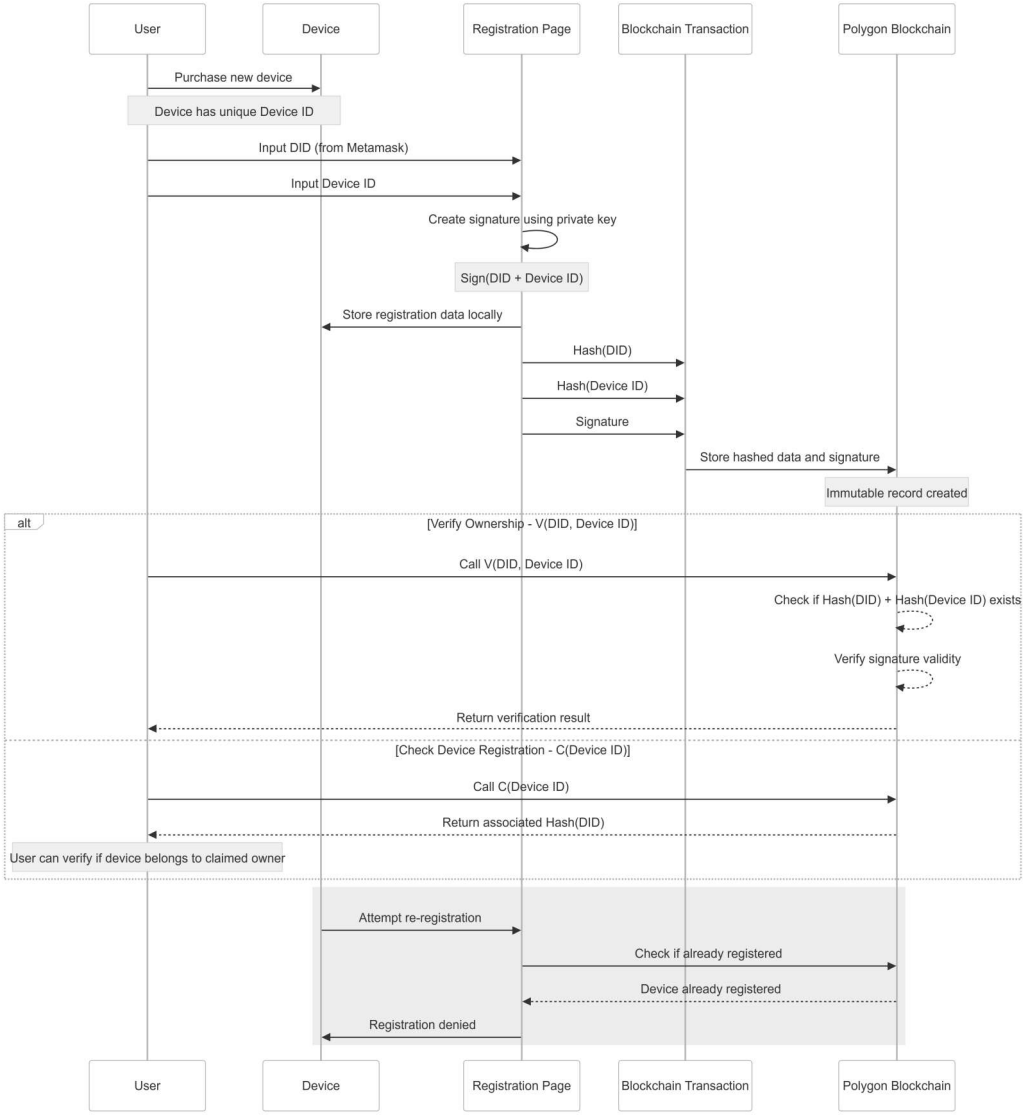

Fig. 3.1 Sequence diagram of the proposed model

3.2 Flowcharts

The following flowcharts detail the key processes of the proposed system:

- **Registration Flowchart**
  This flowchart illustrates how a normal user registers a device using their DID via MetaMask, with signatures stored on the Polygon blockchain.

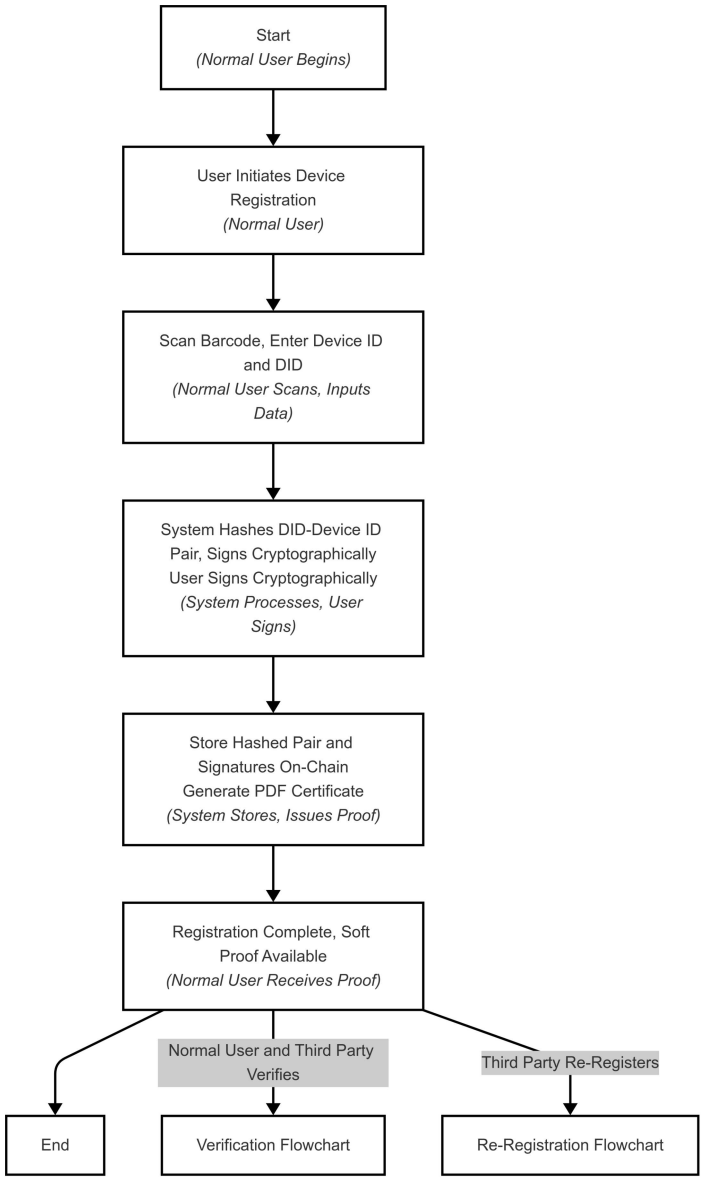

Fig. 3.2 Registartion Flowchart

- **Verification Flowchart**
  This flowchart illustrates the step-by-step process by which a regular user or third parties verifies the ownership of a registered device. This is done by checking the cryptographic signatures and credential records stored securely on the Polygon blockchain.

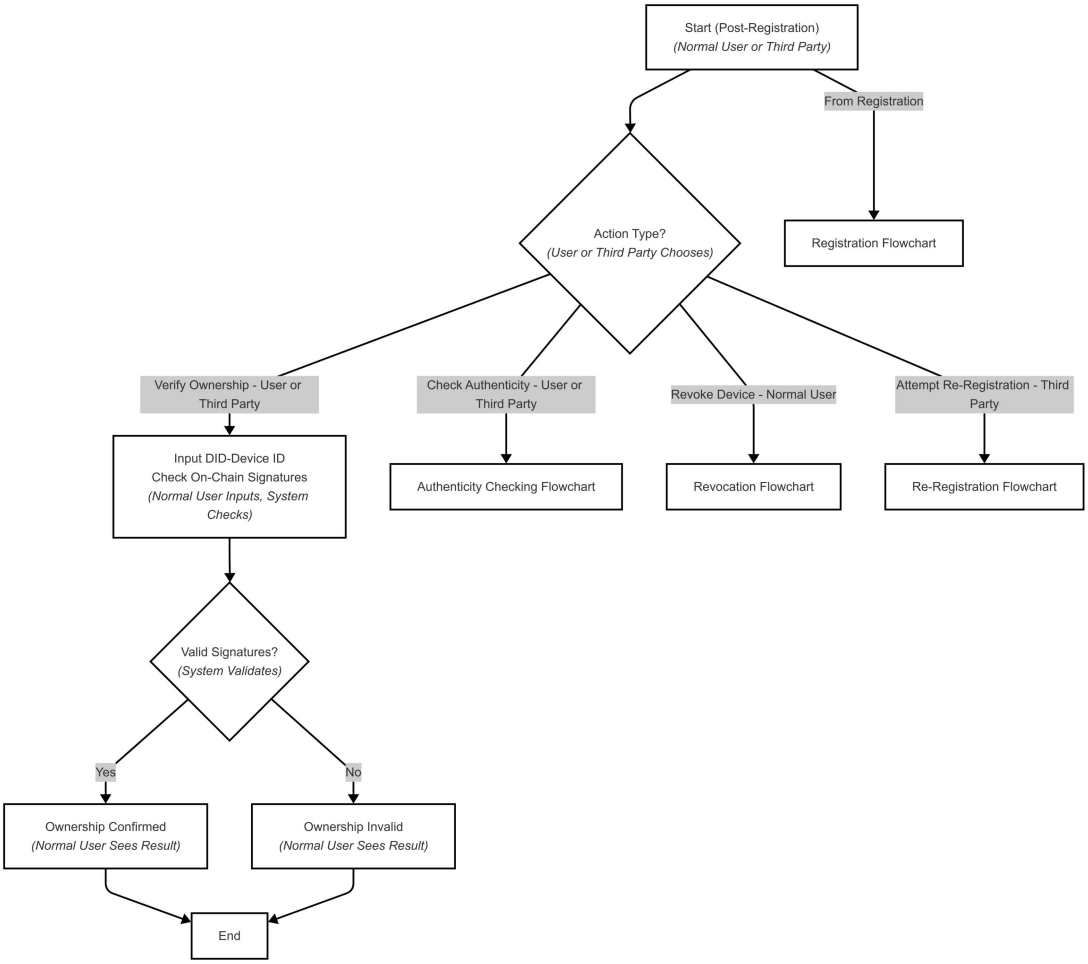

Fig. 3.3 Verification Flowchart

- **Authenticity Checking Flowchart**
  This flowchart depicts how users or third parties check a device's registration status without sensitive data exposure.

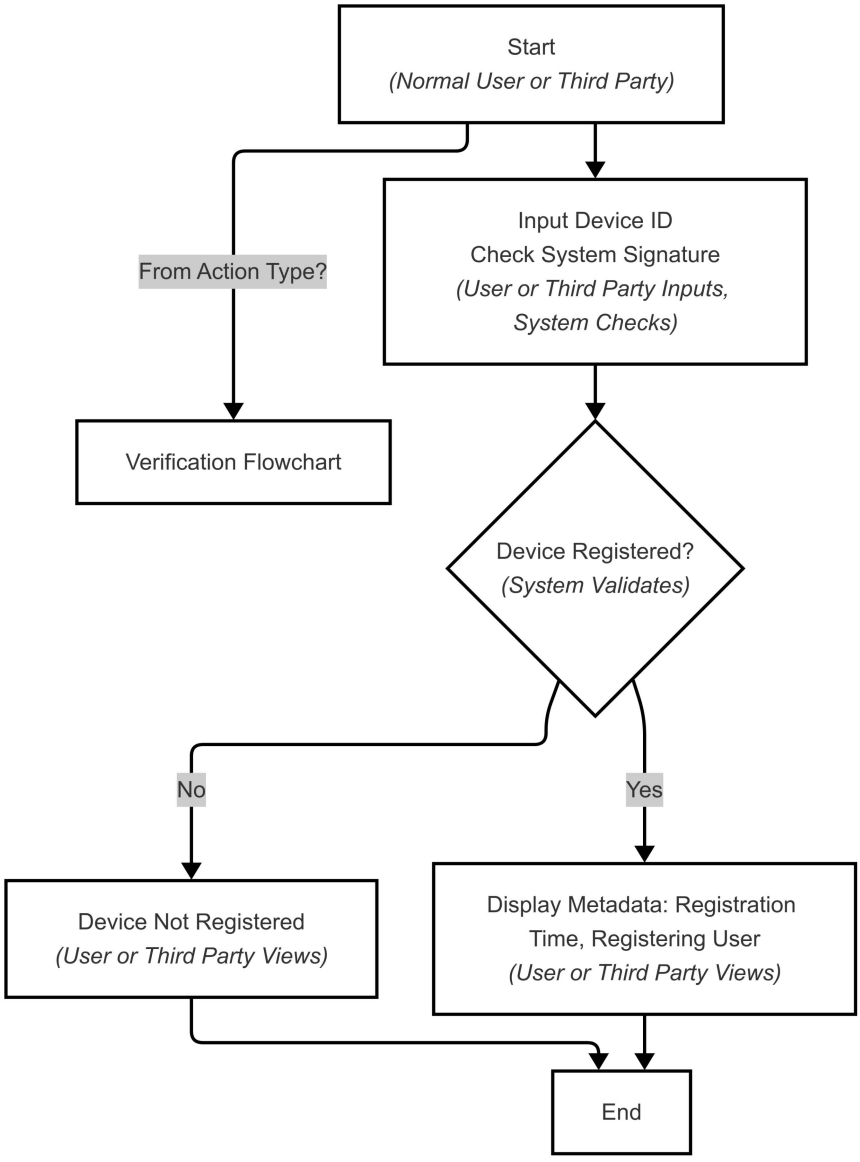

Fig. 3.4 Authenticity Checking Flowchart

- **Revocation Flowchart**
  This flowchart outlines how a normal user revokes a device, blocking it on the blockchain.

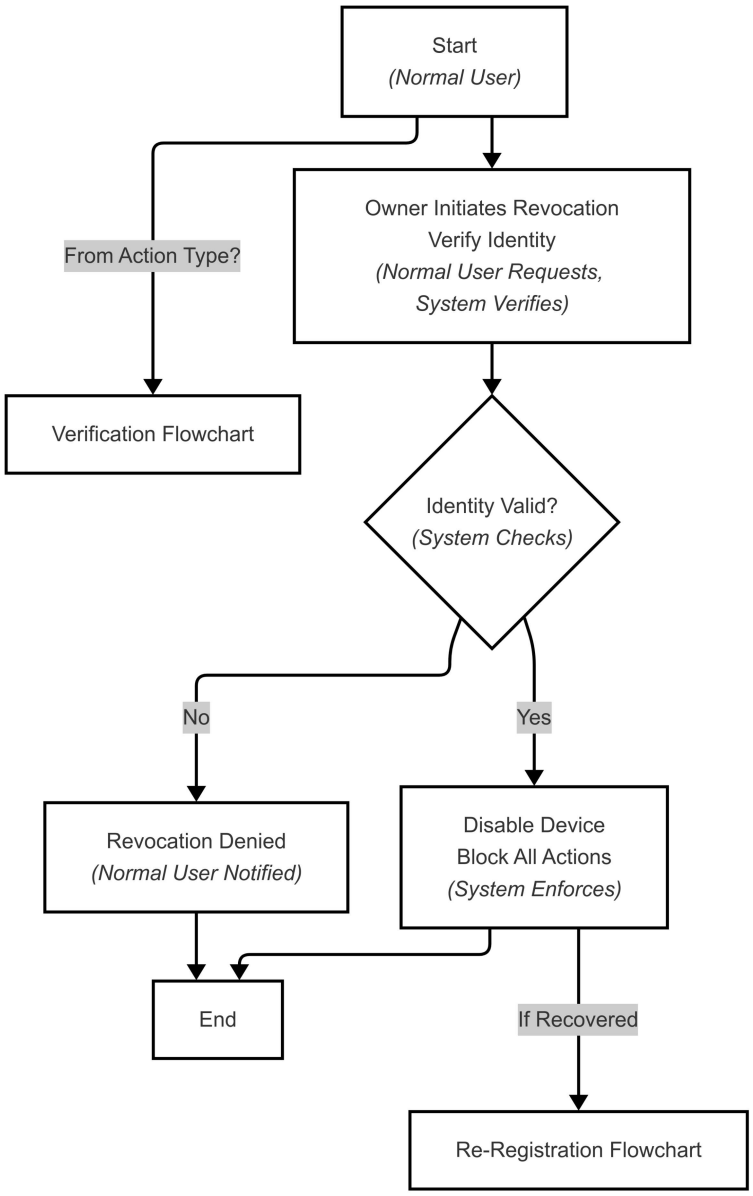

Fig. 3.5 Revocation Flowchart

- **Re-Registration Flowchart**
  This flowchart describes how third-party re-registration attempts trigger alerts and owner recovery processes.

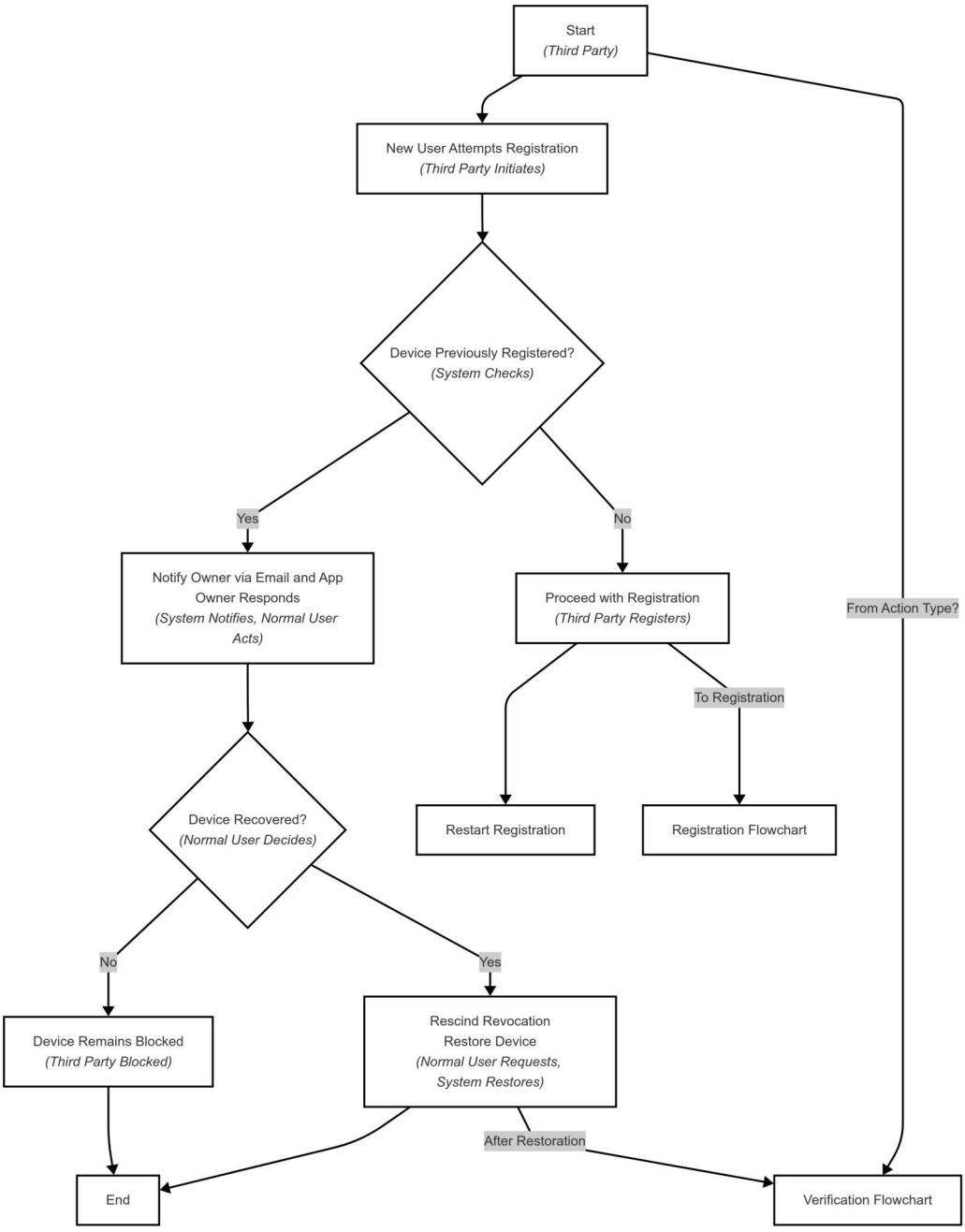

Fig. 3.6 Re-Registration Flowchart

3.3 Algorithms

This section outlines the core algorithms powering the Ownify system: Registration, Verification, Checking, and Revocation/Removal of Revocation. Each algorithm leverages cryptographic primitives within the Ethereum framework, deployed on the Polygon blockchain, to ensure security, integrity, and privacy. The cryptographic operations, including Keccak256 hashing and ECDSA signatures with the secp256k1 curve, are facilitated by the ethers.js library and MetaMask integration, ensuring tamper-proof ownership records and verifiable authenticity.

Cryptographic Mechanisms

The system employs the following cryptographic tools:

- **Keccak256 Hashing**: A SHA-3 variant producing 256-bit hashes, used to anonymize device IDs and DIDs. Implemented via `ethers.keccak256`, it ensures data integrity and consistency with Ethereum smart contracts when combined with solidity-packed encoding (`ethers.solidityPacked`).
- **ECDSA Signatures**: Utilizes the secp256k1 elliptic curve for signing message hashes, executed through `signer.signMessage`. Signatures are verified on-chain using `ecrecover`, linking actions to the signer's Ethereum address for non-repudiation.
- **Polygon Blockchain**: Provides an immutable ledger for storing hashed data and signatures, ensuring tamper resistance and decentralized verification.

These mechanisms collectively secure the system by ensuring that ownership claims are cryptographically signed, verifiable, and resistant to unauthorized modifications.

Algorithm Descriptions

- **Registration**: Registers a device by hashing its ID and the user's DID, signing the combined hash, and recording it on the blockchain, preventing duplicate registrations. For our system, the user's DID is the Metamask Wallet Address of the user.
- **Verification**: Confirms ownership by validating signatures and matching hashed DIDs against blockchain records, ensuring authenticity.

- **Checking**: Queries registration status and ownership details without requiring user signatures, aiding transparency.
- **Revocation/Removal**: Allows the owner to revoke or reinstate a device's status, secured by signature verification and blockchain updates.

Pseudocodes

Algorithm 1 Device Registration

Require: DeviceID, UserDID, Wallet (MetaMask)
Ensure: Device registered on blockchain

$H_{DeviceID} \leftarrow$ Keccak256(toUtf8Bytes($DeviceID$))
$H_{DID} \leftarrow$ Keccak256(toUtf8Bytes($UserDID$))
$Address \leftarrow$ Wallet.getAddress()
$MetaMaskDID \leftarrow Address$
if extractAddress($UserDID$) $\neq$ extractAddress($MetaMaskDID$) **then**
    abort with error "DID mismatch"
end if
$MessageHash \leftarrow$ Keccak256(solidityPacked($H_{DeviceID}, H_{DID}$))
$UserSignature \leftarrow$ ECDSA.sign($MessageHash, Wallet.privateKey$)
$SystemSignature \leftarrow$ ECDSA.sign($MessageHash, System.privateKey$)
$[ExistingH_{DID}, UserSignature, SystemSignature, RegisteredBy, Timestamp, IsRevoked] \leftarrow$ Contract.getRegistration($H_{DeviceID}$)
if $ExistingH_{DID} \neq$ ZeroHash **then**
    if $IsRevoked$ **then**
        abort with error "Device revoked"
    else if $RegisteredBy = Address$ **then**
        abort with error "Already registered"
    else
        Notify original owner and **abort**
    end if
end if
Submit ($DeviceID, UserDID, UserSignature$) to backend for database storage
$Tx \leftarrow$ Contract.registerDevice($H_{DeviceID}, H_{DID}, UserSignature, SystemSignature$)
Wait for Tx confirmation
return success

This algorithm hashes the Device ID and DID, verifies DID ownership, creates user and system signatures, checks for prior registration or revocation, and securely stores the data on-chain, ensuring one-time verifiable device registration.

Algorithm 2 Device Verification

Require: DeviceID, UserDID, Wallet (MetaMask)
Ensure: Ownership verified

$H_{DeviceID} \leftarrow$ Keccak256(toUtf8Bytes($DeviceID$))
$H_{ProvidedDID} \leftarrow$ Keccak256(toUtf8Bytes($UserDID$))
$[H_{DID}, UserSignature, SystemSignature, RegisteredBy, Timestamp, IsRevoked] \leftarrow$ Contract.getRegistration($H_{DeviceID}$)
if $H_{DID} =$ ZeroHash **then**
    return "Device not registered"
end if
if $IsRevoked$ **then**
    abort with error "Device revoked"
end if
$MessageHash \leftarrow$ Keccak256(solidityPacked($H_{DeviceID}, H_{DID}$))
$SystemAddress \leftarrow$ ECDSA.recover($MessageHash, SystemSignature$)
if $SystemAddress \neq SYSTEM\_PUBLIC\_KEY$ **then**
    return "Invalid system signature"
end if
$UserAddress \leftarrow$ ECDSA.recover($MessageHash, UserSignature$)
if $UserAddress \neq$ extractAddress($UserDID$) **then**
    return "User signature mismatch"
end if
if $H_{DID} \neq H_{ProvidedDID}$ **then**
    return "DID mismatch"
end if
return "Ownership verified"

This algorithm checks if a device is registered and not revoked, then verifies the authenticity of both the system and user signatures using ECDSA. It confirms whether the provided DID matches the registered one, ensuring secure and valid ownership verification.

Algorithm 3 Device Checking

Require: DeviceID

Ensure: Registration status returned

$H_{DeviceID} \leftarrow$ Keccak256(toUtf8Bytes($DeviceID$))
$[H_{DID}, UserSignature, SystemSignature, RegisteredBy, Timestamp, IsRevoked] \leftarrow$ Contract.getRegistration($H_{DeviceID}$)
if $H_{DID} =$ ZeroHash **then**
    return "Not registered"
end if
if $IsRevoked$ **then**
    abort with error "Device revoked"
end if
$MessageHash \leftarrow$ Keccak256(solidityPacked($H_{DeviceID}, H_{DID}$))
$SystemAddress \leftarrow$ ECDSA.recover($MessageHash, SystemSignature$)
if $SystemAddress \neq SYSTEM\_PUBLIC\_KEY$ **then**
    abort with error "Invalid signature"
end if
UserDetails $\leftarrow$ Backend.fetchUserDetails(RegisteredBy)
return {isRegistered: true, RegisteredBy, Timestamp, UserDetails}

This algorithm checks if a device is registered and not revoked by hashing the Device ID and fetching registration details from the smart contract. It verifies the system's digital signature using ECDSA to ensure the data's authenticity. If valid, it retrieves user information from the backend using the address that registered the device. Finally, it returns the registration status along with the original registration details.

Algorithm 4 Device Revocation/Removal

Require: DeviceID, UserDID, Wallet (MetaMask), Action ("revoke" or "remove")
Ensure: Device status updated

$H_{DeviceID} \leftarrow$ Keccak256(toUtf8Bytes($DeviceID$))
$[H_{DID}, \_, \_, RegisteredBy, \_, IsRevoked] \leftarrow$ Contract.getRegistration($H_{DeviceID}$)
if $H_{DID} =$ ZeroHash **then**
    abort with error "Device not registered"
end if
if extractAddress($UserDID$) $\neq RegisteredBy$ **then**
    abort with error "Not owner"
end if
if $Action =$ *"revoke"* **then**
    if $IsRevoked$ **then**
        abort with error "Already revoked"
    end if
    $Tx \leftarrow$ Contract.revokeDevice($H_{DeviceID}$)
else
    if $\neg IsRevoked$ **then**
        abort with error "Not revoked"
    end if
    $Tx \leftarrow$ Contract.removeRevocation($H_{DeviceID}$)
end if
Wait for Tx confirmation
return success

This algorithm allows a user to either revoke or remove revocation of a registered device. It first verifies the device is registered and that the requester is the legitimate owner. If the action is "revoke," it marks the device as revoked on-chain, preventing future verification or registration. If the action is "remove," it reactivates a previously revoked device. It ensures proper ownership and status before updating, and confirms the transaction upon success.

3.4 Tech Stacks

The project integrates web, blockchain, and cryptographic technologies to ensure secure and decentralized device identity management. The major components are as follows:

- **Frontend (TypeScript)**: Built using TypeScript, the frontend provides an interactive UI and securely connects to the blockchain using MetaMask, a browser-based Ethereum wallet. MetaMask also serves as the source of the user's Decentralized Identifier (DID) by using the wallet address as their digital identity. For blockchain interaction libraries like ethers.js and web3.js are used.
- **Wallet Integration (MetaMask)**: MetaMask handles identity-related actions like digital signature generation, transaction approval, and address verification, ensuring self-sovereign identity through cryptographic proofs.
- **Backend (Node.js)**: Developed using Node.js, the backend handles business logic, schema validation via Zod, Backend Authentication via JWT, API routing and user data management. It also helps in maintaining off-chain records such as user profiles and registration logs in MongoDB database.
- **Blockchain (Polygon)**: A scalable and low-fee Ethereum-compatible blockchain, is used for decentralized ledger storage. Smart contracts written in Solidity manage device registration, verification, revocation, and ownership checks immutably and transparently.
- **Smart Contract Development**: Solidity is the core programming language for implementing the business logic on-chain. Development and testing are facilitated using Truffle and Ganache. Truffle provides a suite of tools for compiling, deploying, and testing smart contracts, while Ganache allows local blockchain simulation for testing without gas fees.
- **Cryptography**: Keccak256 (Ethereum's standard hashing function) and ECDSA (Elliptic Curve Digital Signature Algorithm) are used for generating and verifying message hashes and signatures. These cryptographic operations are typically facilitated through libraries like ethers.js, web3.js, and Node.js's crypto module.

3.5 Prototype

This prototype implements a secure, decentralized web-based application designed to prove and manage device ownership through Self-Sovereign Identity (SSI) and blockchain technology. Below are the key steps of the system workflow as demonstrated in the interface screenshots, each explaining one major function of the prototype. The cryptographic algorithms and hashing techniques used in this prototype are described in detail in Section 3.3.

3.5.1 User Registration and Login

The process begins when a user visits the website and registers using their essential details such as name, email, password, phone number, and DID (Decentralized Identifier, represented here as a MetaMask wallet address). Once registered, the user logs in and is redirected to the dashboard.

3.5.2 Dashboard Overview

The dashboard provides three main options:
Register New Device – For adding a new device to the system.
Check Ownership – To query the actual owner of a given DID.
Verify Ownership – To validate if a DID is indeed linked to a specific Device ID.

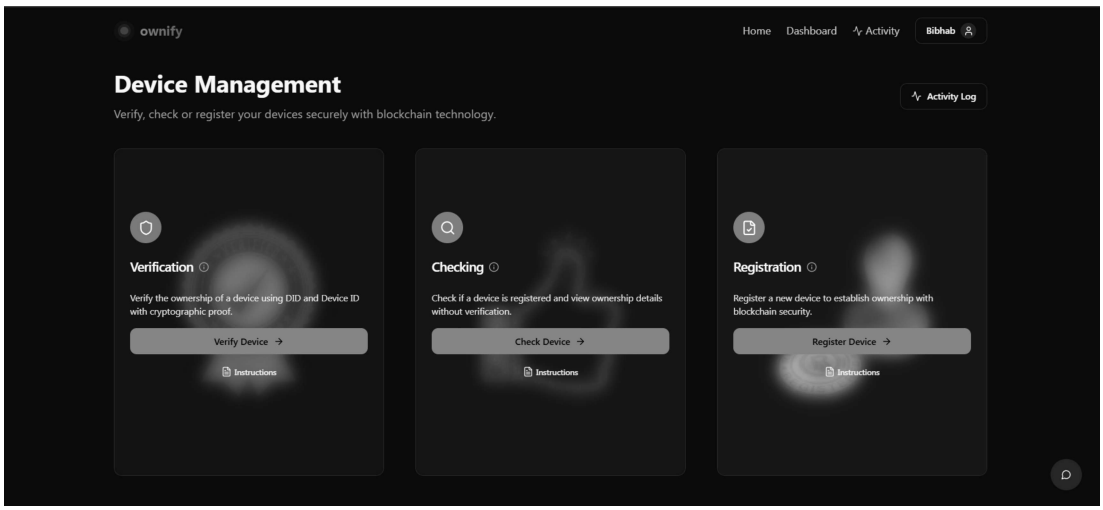

Fig. 3.7 Dashboard

3.5.3 Device Registration

The user enters the DID and Device ID. These are cryptographically signed by both the user's and system's private key to generate verifiable proof. The values are then hashed and stored immutably on the Polygon blockchain. Optionally, users can scan a device barcode to automatically fetch the Device ID – enhancing authenticity. A PDF proof of registration is generated for the user to download and store.

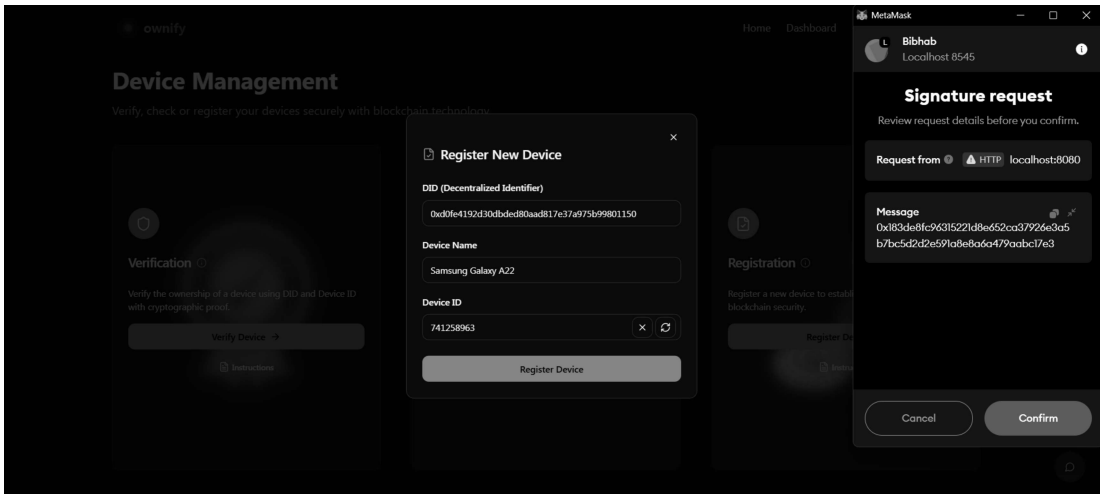

Fig. 3.8 Registration - Entering Details

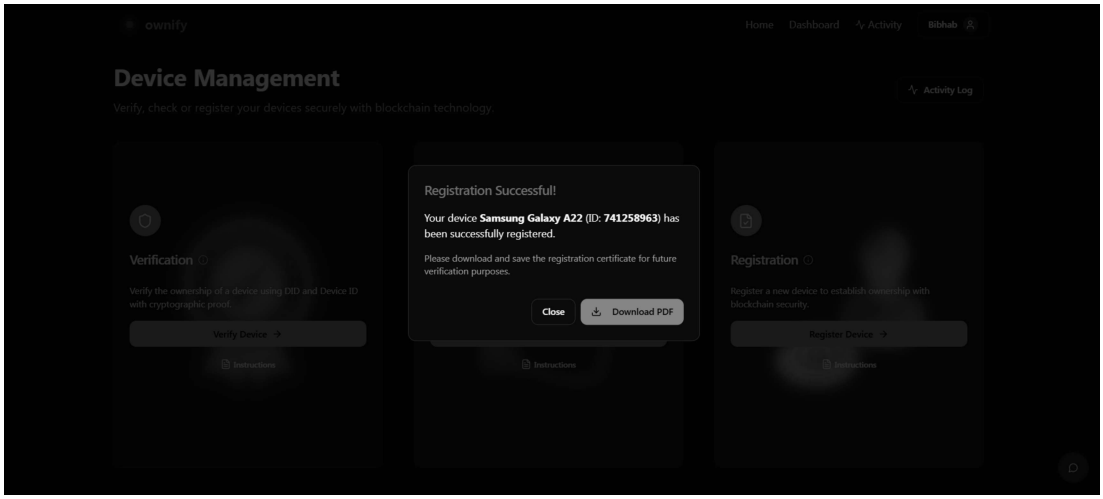

Fig. 3.9 Registration - After Successful Registration

3.5.4 Verification

The verification section requires both DID and Device ID. It checks for the presence of corresponding hashes and validates both user and system digital signatures to confirm true ownership.

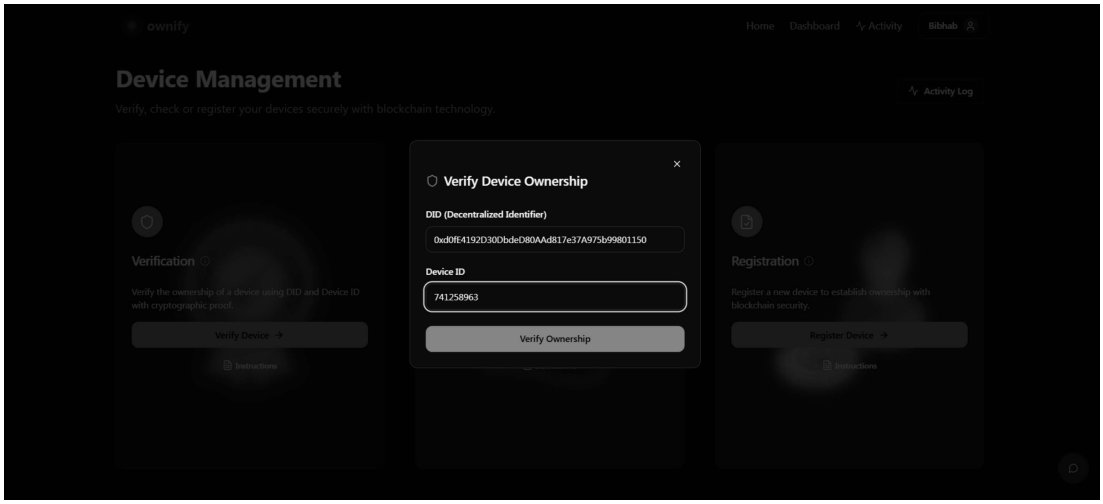

Fig. 3.10 Verification - Entering Details

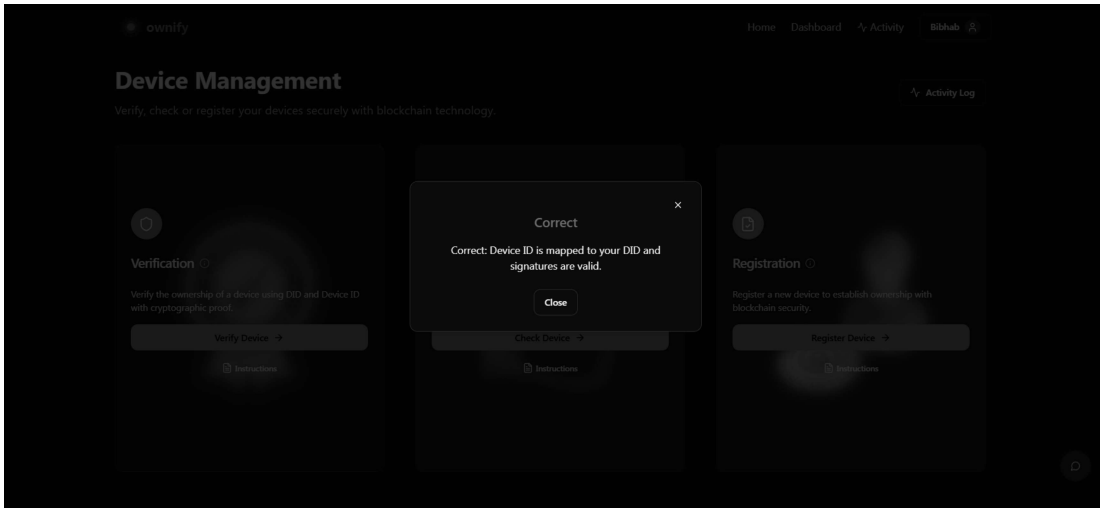

Fig. 3.11 Verification - After Successful Verification

3.5.5 Checking Ownership

This section takes a DID as input and returns the actual registered holder for that device, based on the system's recorded mapping and system signature.

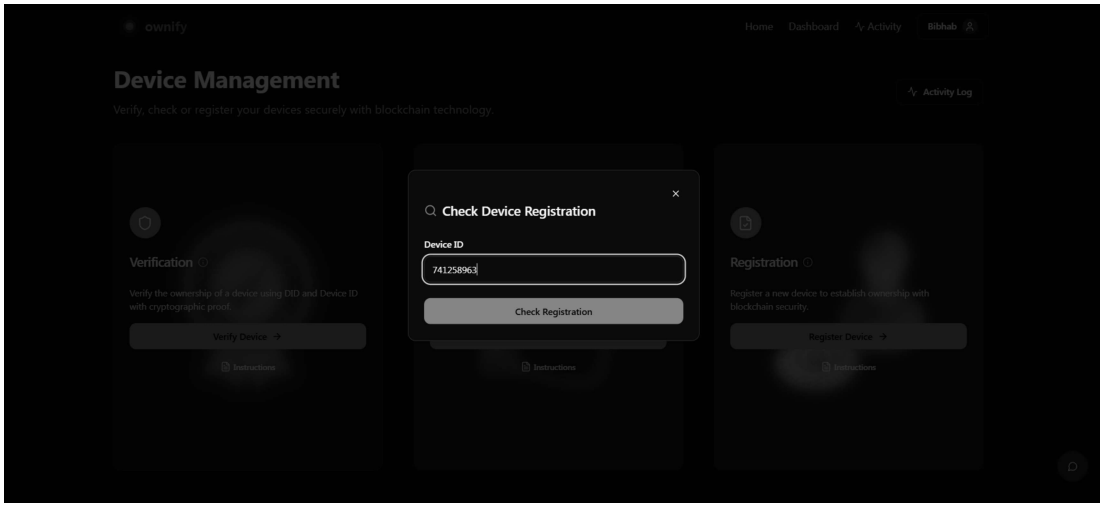

Fig. 3.12 Checking - Entering Details

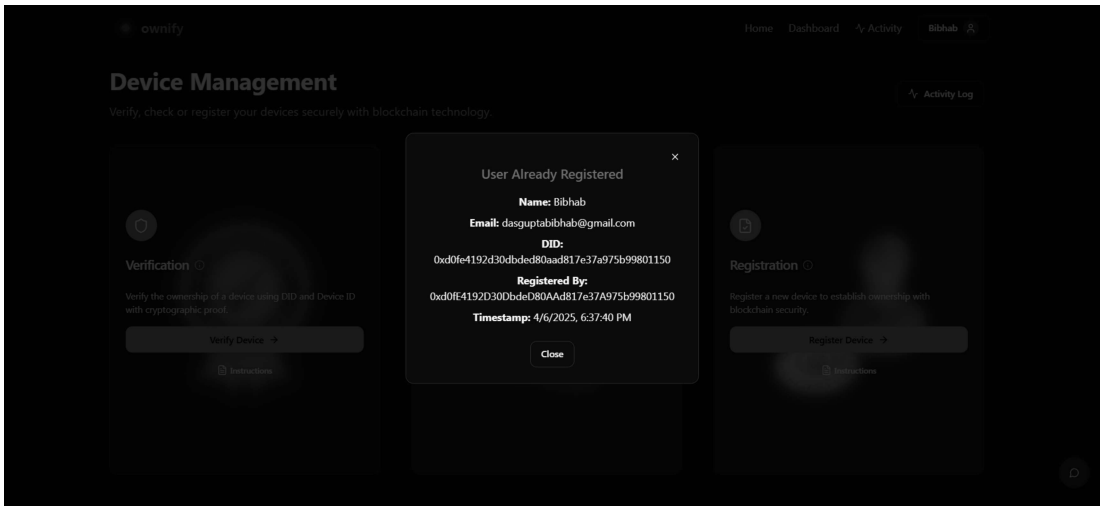

Fig. 3.13 Checking - After Successful Checking

3.5.6 Revocation and Recovery

The original owner can choose to revoke access if their device is lost or stolen. When a device is marked revoked, all re-registration, checking, and verification attempts will show the device as revoked. If recovered, the user can un-revoke the device and restore full rights.

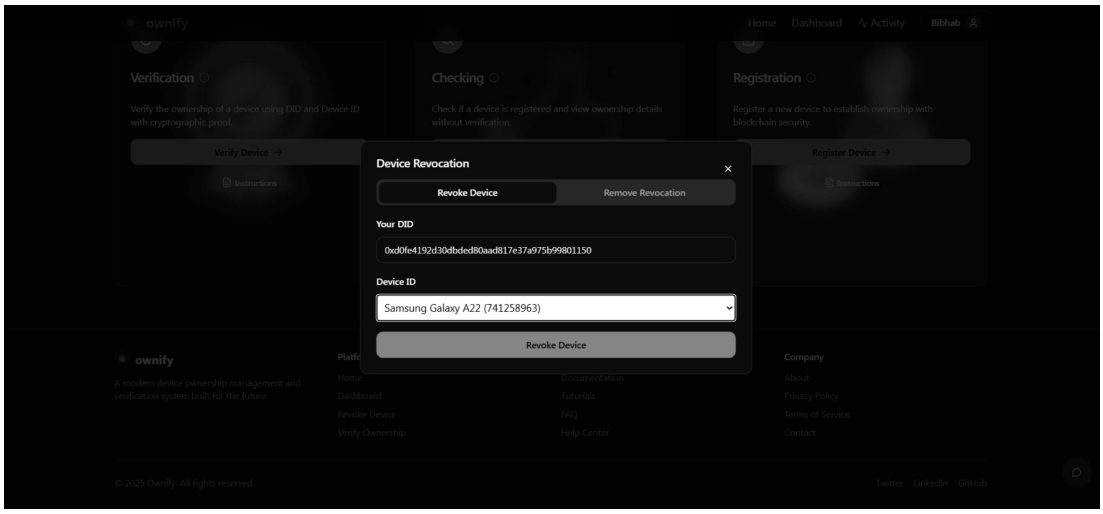

Fig. 3.14 Device Revokation

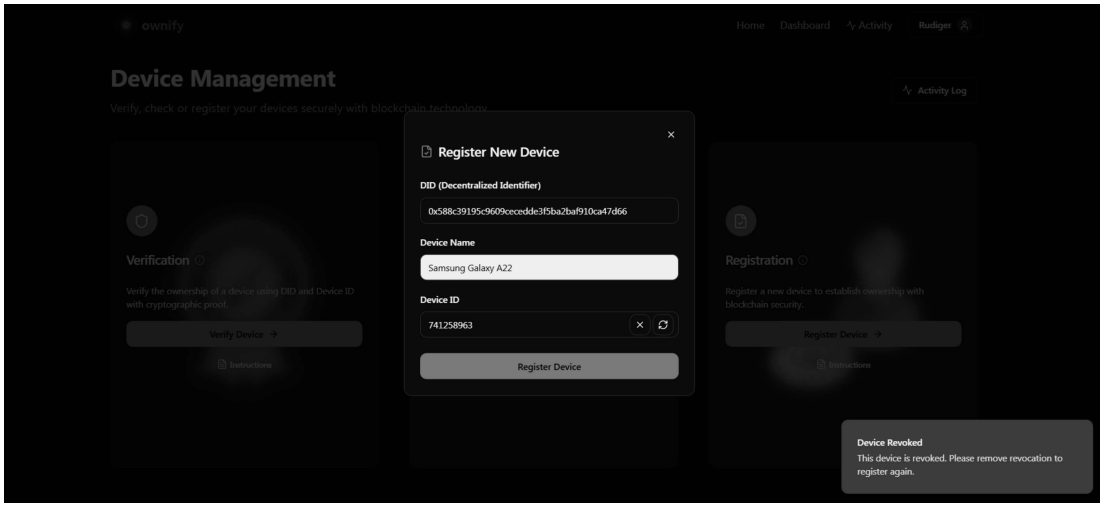

Fig. 3.15 Registering a Revoked Device

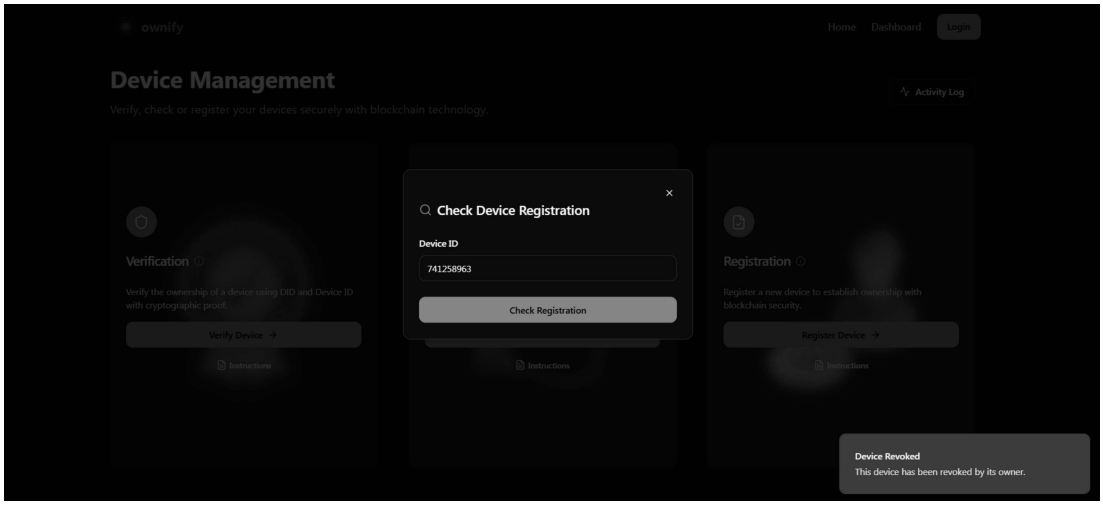

Fig. 3.16 Checking a Revoked Device

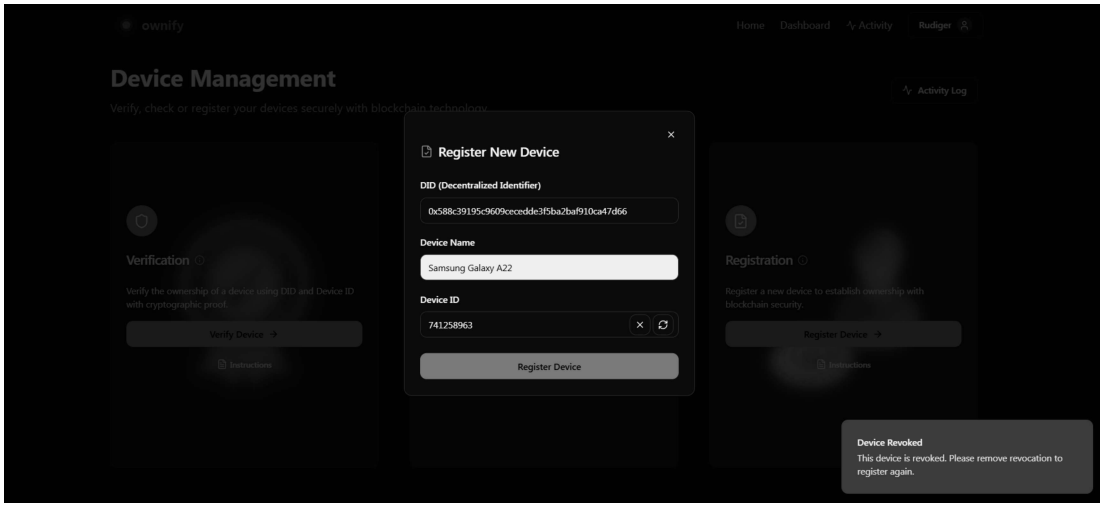

Fig. 3.17 Removing a Revoked Device

3.5.7 Security Alerts and Logs

In the case of unauthorized re-registration attempts, the original user is notified via both email and SMS. The system includes a contact interface to support user concerns, and an activity log for tracking device actions and history.

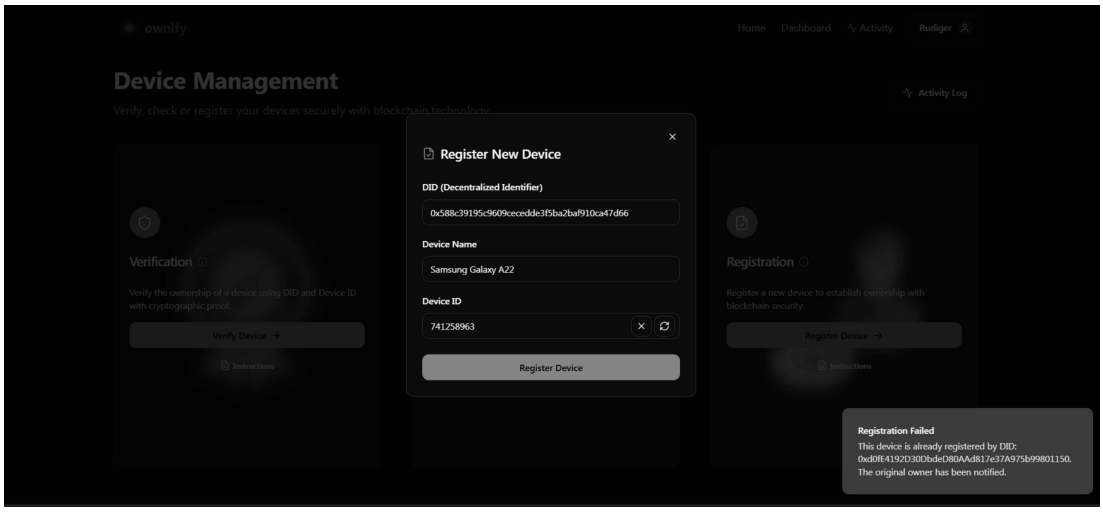

Fig. 3.18 Re-Registration Attempt

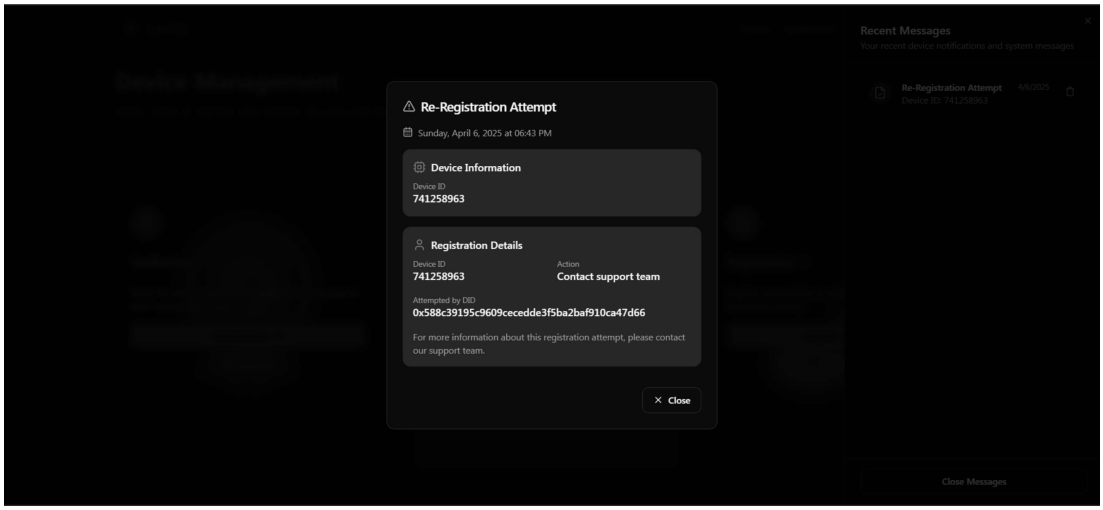

Fig. 3.19 Notifying about the Re-Registration Attempt via message

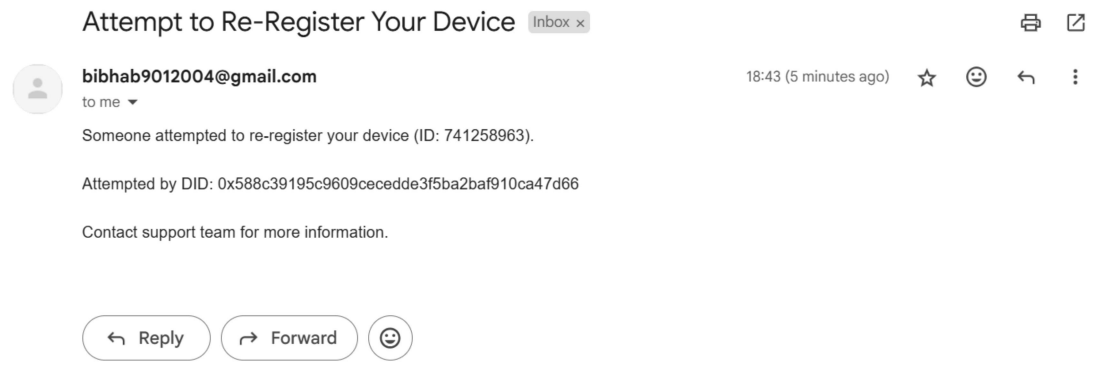

Fig. 3.20 Notifying about the Re-Registration Attempt via email

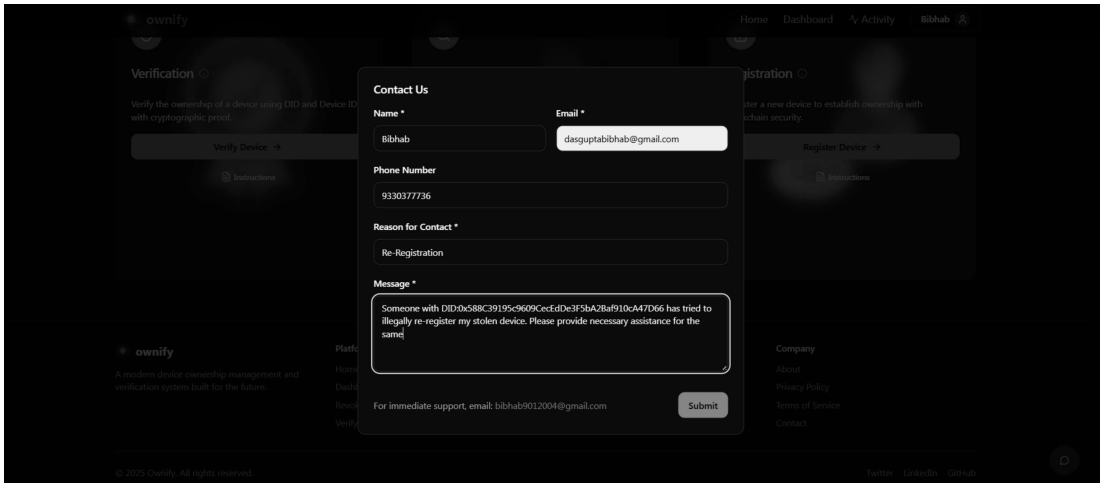

Fig. 3.21 Contact Form for Support

This ensures robust proof of ownership and effectively prevents re-use or sale of black-market devices. The current prototype serves as a strong proof of concept and can be further enhanced by integrating it with hardware-level functionality. In future versions, users may be prompted for verification at startup or at regular intervals, providing additional layers of security similar to biometric and PIN unlocks today.

The hashing and digital signature techniques (user and system based) used during registration ensure that each entry is authentic and non-repudiable. The verification checks both signatures, while the checking function validates the system signature alone to confirm genuine mapping. All implementation details are further elaborated in Section 3.3.

Chapter 4

Conclusion

The project presents a prototype for a decentralized device identity management system that provides tamper-proof and verifiable ownership of electronic devices by means of blockchain and decentralized identifiers. The system generates a tamper-proof record of ownership that cannot be overwritten or re-registered, which will effectively deter unauthorized resale or placement of the device on the black market by binding a user's DID (here, the MetaMask wallet address) with a device's unique identifier (say, IMEI, serial number, or UUID).

4.1 Brief Overview

The prototype offers an intuitive user interface with essential features like device registration, ownership verification, authenticity checking, and revocation handling. During registration, the system securely hashes and stores the DID-device ID pair, along with cryptographic signatures from both the user and the system, which are later used to verify the integrity and authenticity of ownership claims. A downloadable PDF certificate is generated as soft proof of ownership. To ensure physical possession, the system supports barcode scanning during registration. The verification function validates a DID-device ID link by checking both user and system signatures stored on-chain, confirming the legitimacy of ownership. Meanwhile, the checking function allows users or third parties to confirm whether a device is registered and view limited non-sensitive metadata—like registration time and registering user—using only the system signature. This provides a quick and privacy-preserving integrity check without exposing sensitive ownership data.

In the case of theft or loss, a strong revocation mechanism exists in the system whereby the original owner can disable a device to make it completely unusable in the ecosystem. This blocks all verification, checking, or re-registration attempts and enforces a strict one-time registration policy against black market resale or unauthorized reuse. If the owner forgets to revoke the device and someone else attempts to re-register, prompt email notification and in-app alert enable quick response. These alerts, together with a built-in contact feature, therefore, constitute a critical first layer of defense. Once the device is located, the revocation can be rescinded and the device restored without re-registration. Every user actions, be it registration, verification, checking, or revocation, is permanently recorded, with real-time updates ensuring transparent, auditable device history, increased full life cycle control.

4.2 Future Prospects

Despite being well-designed, there are several challenges. For large-scale deployment, integration with existing hardware and legacy systems will be paramount. The integration of the mobile OS also allows for tighter coupling, involving verification of devices during boot time or at intervals, similar to PIN- or biometric-based locks, thus embedding security at a very fundamental level. Manufacturers and software vendors will have to participate. Another challenge includes ensuring interoperability with existing or unsupported devices and with guaranteed privacy for users as per data protection regulations like GDPR. A balance must be struck between decentralization and safeguards on ethical grounds against misuse or surveillance, especially when linking sensitive identity information with physical assets.

On the contrary, the remedy provides enormous benefits: transparency, verifiability, and trust devoid of intermediaries, making room for tamper-proof activity logs, secure communication with support, and enhanced sovereignty over actions for users. In its improvement trajectory, a multi-blockchain solution with enhanced UI/UX for easy access, biometric integration for extra safeguarding, and enhanced performance on low-configuration devices may be considered. Furthermore, collaborative work with OEMs and regulators is to cement the acceptance of the system as a standard. The project offers clear evidence, not merely as a proof of concept, of the technical and practical feasibility of a safe, secure and reliable self-sovereign device identity management system.

References

[1] Ahmed, M. R., Islam, A. K. M. M., Shatabda, S., and Islam, S. (2023). Blockchain-based identity management system and self-sovereign identity ecosystem: A comprehensive survey. In *Proceedings of the IEEE Conference on Blockchain and Identity Management*. IEEE. 1, 6

[2] Bai, Y., Gao, H., Lei, H., Li, J., Li, S., and Li, L. (2022). Decentralized and self-sovereign identity in the era of blockchain: A survey. In *IEEE International Conference on Blockchain (Blockchain)*. 6

[3] Chen, Y., Liu, C., Wang, Y., and Wang, Y. (2023). A self-sovereign decentralized identity platform based on blockchain. In *Proceedings of the Institute of Information Engineering*. Chinese Academy of Sciences. 2, 5

[4] Eddine, B. N., Ouaddah, A., and Mezrioui, A. (2021). Exploring blockchain-based self sovereign identity systems: Challenges and comparative analysis. In *2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, Rabat, Morocco. IEEE. 7

[5] Ferdous, M. D. S., Chowdhury, F., and Alassafi, M. O. (2023). In search of self-sovereign identity leveraging blockchain technology. In *Proceedings of the Department of Computer Science and Engineering*. Shahjalal University of Science and Technology. 6

[6] Jensen, J. (2012). Federated identity management challenges. Master's thesis, Norwegian University of Science and Technology (NTNU), Trondheim, Norway. 5

[7] Jiang, J., Duan, H., Lin, T., Qin, F., and Zhang, H. (2023). A federated identity management system with centralized trust and unified single sign-on. In *Proceedings of the Department of Computer Science and Technology*. Tsinghua University. 4

[8] Kaneriya, J. and Patel, H. (2020). A comparative survey on blockchain-based self sovereign identity system. In *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, Gandhinagar, India. IEEE. 1, 7

[9] Koradia, D. and Agrawal, V. (2024). Study of self-sovereign identity management system incorporating blockchain. *International Journal of Intelligent Systems and Applications in Engineering*, 12(22s):83–91. Available at https://www.ijisae.org. 7

[10] Maler, E. and Reed, D. (2008). The venn of identity: Options and issues in federated identity management. Sun Microsystems and Cordance. 1, 5

[11] Naicker, D. D. and Moodley, M. M. (2024). Pragmatics behind idealistic digital identity management. Details available upon request or pending publication. 1, 5

[12] Naik, N. and Jenkins, P. (2021a). Sovrin network for decentralized digital identity: Analysing a self-sovereign identity system based on distributed ledger technology. In *2021 IEEE International Symposium on Systems Engineering (ISSE)*. IEEE. 7

[13] Naik, N. and Jenkins, P. (2021b). uport open-source identity management system: An assessment of self-sovereign identity and user-centric data platform built on blockchain. In *Proceedings of the School of Informatics and Digital Engineering*. Aston University and Cardiff Metropolitan University. 7

[14] Naik, N. and Jenkins, P. (2022). Governing principles of self-sovereign identity applied to blockchain enabled privacy preserving identity management systems. *International Journal of Advanced Computer Science and Applications*. 7

[15] Niemiec, M. and Kolucka-Szypuła, W. (2015). Federated identity in real-life applications. *AGH University of Science and Technology*. 4

[16] Omar, A. S. and Basir, O. (2020). Decentralized identifiers and verifiable credentials for smartphone anticounterfeiting and decentralized imei database. *Journal of Information Security and Applications*. 2

[17] Satybaldy, A., Ferdous, M. S., and Nowostawski, M. (2022). A taxonomy of challenges for self-sovereign identity systems. *IEEE Access*, 10. 2, 6

[18] Schardong, F. and Custódio, R. (2024). Self-sovereign identity: A systematic review, mapping and taxonomy. 5

[19] Shin, D., Ahn, G.-J., and Shenoy, P. (2004). Ensuring information assurance in federated identity management. In *Laboratory of Information Integration, Security, and Privacy (LIISP)*. University of North Carolina at Charlotte. 4, 5

[20] Soltani, R., Nguyen, U. T., and An, A. (2023). A survey of self-sovereign identity ecosystem. In *Proceedings of the Lassonde School of Engineering*. York University. 2, 6

[21] Špela Čučko, Beširović, S., Kamisalić, A., Mrđović, S., and Turkanović, M. (2022). Towards the classification of self-sovereign identity properties. *IEEE Access*. 5