Interreg
Euregio Meuse-Rhine
BC4P
EUROPEAN UNION
European Regional
Development Fund

A PROTOTYPE INTERFACE FOR THE BLOCKCHAIN

(using decentralized attribute based identity management)
Deliverable DT3.1.1.

Version	Description	Author	Date
0.1	Initial draft	Max Smeets (Zuyd NL), Prakash Gupta (OUNL), Pietro Piccini (OUNL), Stefano Bromuri (OUNL),	11/01/2023
0.2	Second draft	Prakash Gupta, Pietro Piccini, Stefano Bromuri	15/02/2023
1.0	Final version	Prakash Gupta, Pietro Piccini, Stefano Bromuri	28/02/2023

PXL
LIÈGE université
FH Aachen
University of Applied Sciences
Open Universiteit
howest
Tiorc
ZU YD
CLIMATE CITIES

Table of Contents

ABSTRACT **3**
1. INTRODUCTION **4**
2. BACKGROUND **5**
2.1 OpenZeppelin and NFTs 5
2.2 Decentralized Identity Management Systems (dIMS) 7
2.3 Zero-Knowledge Proofs 7
3. THE ARCHITECTURE OF THE BC4P DIGITAL IDENTITIES SOLUTION **8**
3.1 IERC721 Interface for Interface NFTs 8
3.2 SoulBoundToken 10
3.3 Example 11
3.4 Minting Process 13
3.5 Transferring Tokens 14
4. CONCLUSION AND FUTURE WORK **17**
5. REFERENCES **17**

Abstract

Rising energy costs have become a significant challenge for low and middle-income consumers. The Blockchain 4 Prosumers project proposes a decentralized, peer-to-peer energy market that leverages blockchain technology to eliminate intermediaries and offer competitive prices. However, implementing this solution requires a secure and privacy-compliant identification system for participants. The project has developed a decentralized and privacy-friendly identity management system for users and their assets to overcome this challenge. This project's objectives, challenges, and solutions in implementing a privacy-sensitive for the blockchain-based energy market.

1. Introduction

The current energy crisis has resulted in a significant increase in energy prices, as indicated by figure 1, which demonstrates the correlation between a google trends analysis on “placing solar panels” and the average energy price in kw/h.

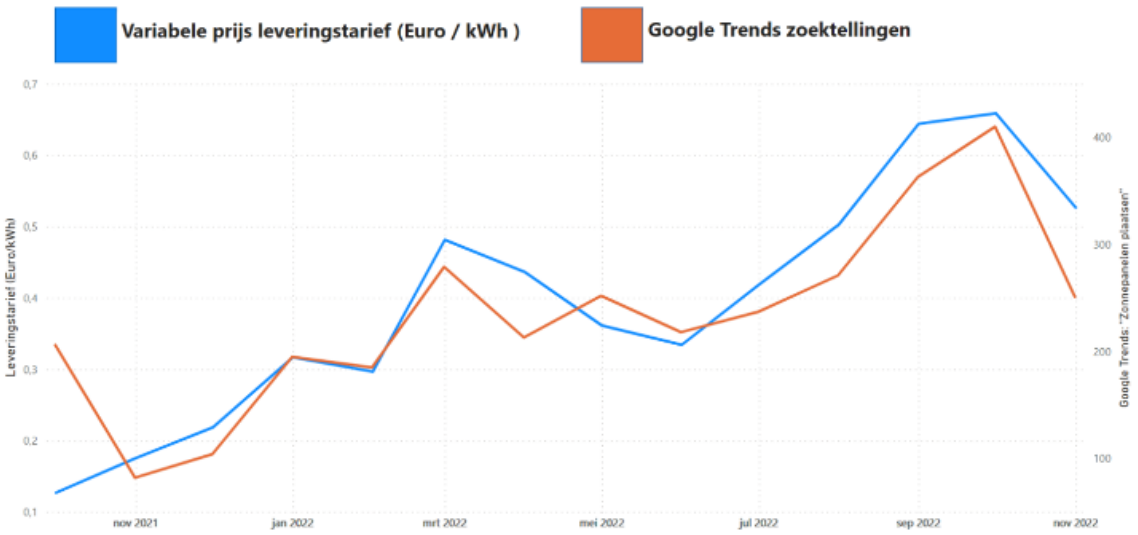


Figure 1: Trend analysis of "placing solar panels" compared to kw/h price.

The planned phasing exacerbates this crisis out of the existing net metering scheme by the Dutch government, which is expected to occur between 2025 and 2031 (1). This will reduce compensation for residual energy flow, ultimately leading to a zero return and prolonging the payback period for solar panels This highlights the need for an alternative solution.

The objective of BC4P[2] is to integrate blockchain technology[15] with the prosumer to establish local energy markets within the EMR through peer-to-peer trading. In addition, the project aims to educate various stakeholders, including governments, companies, prosumers, and consumers, about the benefits of blockchain technology.

Blockchain technology is a versatile innovation that has been utilized in a wide range of use cases. Mainly it enables peer-to-peer transfer of value over a computer network without the need for third-party intermediaries who serve as a layer of trust between the parties involved. Beyond trading cryptocurrencies such as Bitcoin, blockchain technology enables the digitization and trading of other physical goods as well. In the context of the energy market, this technology has the potential to eliminate the involvement of a network operator in the trading of residual current. However, for the efficient functioning of local energy markets enabled by blockchain technology, it is imperative that participants are able to identify each

other. Currently, this could be achieved through conventional know-your-customer (KYC) methodologies. However, recent advancements in blockchain technology provide an opportunity to digitize and decentralize these processes.
The objective is to create mutual identification among entities within the peer-to-peer energy market. This artifact will integrate decentralization and privacy and serve as the foundation for business within the peer-to-peer energy market with identified entities.

The main contribution of this deliverable is, therefore, to provide smart contracts to manage digital identities in a blockchain solution, and it builds directly on top of the WP2.T2 deliverable of the BC4P project.

Specifically, the rest of this document is structured as follows: the background Section discusses blockchain concepts that are relevant to understand the digital identifies solution proposed in this document; the architecture Section describes our proposition to solve the digital identifies manages; finally, the conclusion Section concludes this report and discusses potential future work.

2. Background

This Section discusses the relevant technological background necessary to understand how the smart contracts of the BC4P project manage digital identities. As such, BC4P uses non-fungible tokens (NFT) [3]to manage digital identities in a decentralized manner. NFTs are defined in the OpenZeppelin standard.

2.1 OpenZeppelin and NFTs

OpenZeppelin[4] is an open-source library of smart contracts and tools for building decentralized applications (dApps)[5] on the Ethereum blockchain. The project was launched in 2016 and has since become one of the most widely used libraries in the Ethereum ecosystem. The library provides developers with a secure and audited framework for building smart contracts, allowing them to focus on building the functionality of their dApps.

The primary goal of OpenZeppelin is to enhance the security of smart contracts on the Ethereum blockchain by providing developers with a standardized set of building blocks that they can use to create secure, reliable, and efficient dApps. The library includes many smart contract templates, such as ERC20[6], ERC721[7], and ERC777[8] tokens, and other useful contacts like multi-signature wallets, access control, and upgradeable contracts.

One of the primary advantages of OpenZeppelin is that it has been extensively audited by security experts, making it one of the most secure and reliable libraries available for Ethereum development. The library has been reviewed by several security firms, including

Trail of Bits, ChainSecurity, and ConsenSys Diligence, and has undergone multiple iterations to improve its security and reliability.

OpenZeppelin's smart contract templates are designed to be modular, so developers can easily customize them to fit their specific needs. This modular approach makes it easy to create custom contracts tailored to the unique requirements of each dApp, while benefiting from the security and reliability of the OpenZeppelin library.

In essence, NFTs for the Ethereum blockchain is specified as an extension of the ERC721 contract. Non-fungible tokens (NFTs) have become a popular topic recently with the rise of blockchain technology and cryptocurrency. NFTs are unique digital assets stored on a blockchain, providing a decentralized, immutable record of ownership. With the integration of decentralized identity management systems (dIMS) and the use of the IERC721 interface for interface NFTs, NFTs can provide various benefits and opportunities for creators and collectors.

ERC721 is a standard for non-fungible tokens (NFTs) on the Ethereum blockchain. It was introduced in 2018 to create unique, indivisible, and non-interchangeable tokens that could represent a wide range of digital assets.

Unlike other token standards on Ethereum, such as ERC20, which are fungible and can be exchanged for each other, ERC721 tokens are non-fungible, meaning that each token is unique and cannot be exchanged for another token. This makes ERC721 tokens ideal for representing unique assets like artwork, collectibles, and in-game items.

One of the primary benefits of ERC721 is that it provides a standardized way of creating and managing NFTs. This standardization makes it easier for developers to create and integrate NFTs into their dApps and allows for interoperability between different platforms that support ERC721 tokens.

Another benefit of ERC721 is that it allows for the ownership and provenance of digital assets to be tracked on the blockchain. Since each ERC721 token is unique and cannot be exchanged for another token, it can be used to represent a specific digital asset and ensure that the ownership of that asset is securely recorded on the blockchain.

ERC721 has become an essential standard for creating and managing NFTs on the Ethereum blockchain. It has enabled many use cases, from artwork and collectibles to in-game items and more. With ERC721, developers have a standardized way of creating and managing NFTs, ensuring interoperability and enabling the secure ownership and provenance of digital assets.

2.2 Decentralized Identity Management Systems (dIMS)

Decentralized identity management systems (dIMS)[9] are blockchain-based systems that enable individuals to control their digital identities, including personal information and online activities, without the need for a centralized authority. This system allows users greater security and privacy, as they have full control over their data. It eliminates the risk of data breaches or hacks that occur on centralized servers.

In the context of NFTs, dIMS can be used to establish the unique identity of each NFT and link it to its owner. This system allows for secure, verifiable ownership of the NFT, ensuring that the original creator or current owner is recognized as the legitimate owner of the asset. Additionally, dIMS can provide opportunities for creators to build their brand and reputation within the NFT community by establishing a verifiable track record of their past works.

2.3 Zero-Knowledge Proofs

Zero-knowledge proofs (ZKPs)[10] are a cryptographic technique that allows a prover to demonstrate knowledge of a secret without revealing any information about that secret to a verifier. This technique has many applications, including in the field of blockchain technology and the creation of non-fungible tokens (NFTs).

In the blockchain context, ZKPs can enhance privacy and security by allowing users to prove ownership or control of certain assets without revealing sensitive information about themselves or the assets. For example, ZKPs can be used to prove ownership of a private key without revealing the key itself or to prove that a transaction is valid without revealing any details.

NFTs are a unique digital asset that cannot be exchanged on a one-for-one basis like traditional currencies or cryptocurrencies. ZKPs can enhance the security and trustworthiness of NFTs by allowing for the creation of provably authentic and unique tokens. By using ZKPs to prove the authenticity and uniqueness of an NFT, it becomes much more difficult for fraudsters to create fake tokens or counterfeit existing ones.

A prover can demonstrate that they know a certain secret without revealing any information about that secret to the verifier. This can prove the uniqueness and authenticity of an NFT by demonstrating that the token was created in a certain way without revealing any information about the process used to create it.

Overall, ZKPs is a powerful tool in blockchain technology and NFTs. By creating provably authentic and unique digital assets, ZKPs can help enhance security, trustworthiness, and privacy in these emerging fields.

3. The architecture of the BC4P Digital Identities Solution

In decentralized identity management, blockchain technology and NFTs provide a platform for digitizing and storing proof of identity in a secure and tamper-resistant manner. By leveraging the concepts of decentralization and immutability, blockchain technology offers a foundation for creating a decentralized identity management system (dIMS) that can serve as the foundation for peer-to-peer interactions within various domains, including the energy market.

There are, however, some problems with the current state of basic NFTs, which make them unsuited for a dIMS. Firstly, these NFTs should not be tradeable within the blockchain network. To combat this, they can be made “soulbound”. This means that once the NFT is minted on-chain, it is bound to the wallet in which it is minted [11]. Secondly, there is the problem that NFT metadata is generally static [12]. Within the world of digital art, this static metadata makes sense. However, within a dIMS, personal documents change all the time, so metadata should be dynamic. The metadata should be integrated within the contract to avoid working with centralized and decentralized storage. To make this metadata updateable, a function can be implemented to update this metadata under certain conditions. This is called a dynamic NFT (dNFT) [13]. One of these conditions would be that only the token owner can trigger this function.

3.1 IERC721 Interface for Interface NFTs

As specified in the background Section, the IERC721 interface is a standard interface for non-fungible tokens, and it provides a framework for NFTs to be created, traded, and managed on the blockchain. Interface NFTs are a specific type of NFT designed to represent a unique user interface or application interface rather than a physical or digital asset.

Interface NFTs can provide a range of benefits for developers and users. For developers, interface NFTs can serve as a form of payment for their work, ensuring they receive recognition and compensation for their contributions. For users, interface NFTs can provide a unique, personalized experience for their online activities and can serve as a form of digital identity or reputation.

With the integration of dIMS and the use of the IERC721 interface for interface NFTs, creators, and collectors can benefit from increased security, verifiability, and ownership of their digital assets. NFTs can provide a new avenue for creators to monetize their work and build their brand within the community. At the same time, collectors can enjoy the unique ownership experience and potential financial benefits of owning rare or valuable NFTs.

A soulbound token[14] is a non-fungible (NFT)[14] designed to attach to a specific user's account or wallet permanently. This attachment makes the NFT unique to the user and ensures that it cannot be transferred or sold to another user. Soulbound tokens are important for NFTs because they ensure that a digital asset's ownership and authenticity are maintained.

```solidity
function _mint(address to, uint256 tokenId) internal virtual {
    require(to != address(0), "ERC721: mint to the zero address");
    require(!_exists(tokenId), "ERC721: token already minted");

    _beforeTokenTransfer(address(0), to, tokenId, 1);

    // Check that tokenId was not minted by `_beforeTokenTransfer` hook
    require(!_exists(tokenId), "ERC721: token already minted");

    unchecked {
        // Will not overflow unless all 2**256 token ids are minted to the same owner.
        // Given that tokens are minted one by one, it is impossible in practice that
        // this ever happens. Might change if we allow batch minting.
        // The ERC fails to describe this case.
        _balances[to] += 1;
    }

    _owners[tokenId] = to;

    emit Transfer(address(0), to, tokenId);

    _afterTokenTransfer(address(0), to, tokenId, 1);
}
```

Figure 2: Function to mint a token.

When an NFT is minted as a soulbound token, it becomes uniquely tied to a specific user's account or wallet. This means the NFT cannot be duplicated or transferred to another user, making it more secure and less susceptible to fraud.

Soulbound tokens are useful for NFTs representing unique, high-value assets, such as artwork or collectibles. Attaching an NFT to a specific user's account becomes a more valuable and secure asset since the ownership and authenticity of the asset are guaranteed. This can also help prevent counterfeiting, as anyone with access to the blockchain can easily verify the NFT's ownership and authenticity.

Overall, soulbond tokens are an important tool for ensuring the security and authenticity of NFTs. Tying an NFT to a specific user's account or wallet becomes a more valuable and secure asset, particularly important for high-value digital assets like artwork and collectibles.

3.2 SoulBoundToken

The technical implementation of NFTs is a series of solidity interfaces that can be implemented into smart contracts. In solidify, an interface is a class that only contains abstract functions. Every contract that extends an interface has to provide an implementation for the functions defined by the interface. The most common interface which is used for NFTs is the IERC721 interface.

More often than not, developers extend the functionality of the ERC721 smart contract, which is an implementation of the IERC721 interface that contains minting functionalities, which are common for most NFT use cases.

The SoulBoundToken NFT is a further extension of the ERC721 contract, which has functionalities for reading, uploading, and updating metadata as well as blocking any NFT transfer attempt.

SoulBound Tokens and assets are a significant aspect of the blockchain ecosystem, providing users with an unprecedented level of transparency and security. By linking these assets to a user's Ethereum account, the system creates a unique and unchangeable identity for each asset, making it nearly impossible for it to be lost, stolen, or duplicated.

The process of linking assets and tokens to a user's Ethereum account is simple yet powerful. When a user logs into the interface with Metamask, the system detects the public key associated with their Ethereum account. This public key is then used to create the assets and SoulBoundToken associated with that account whenever the user performs an operation through the interface.

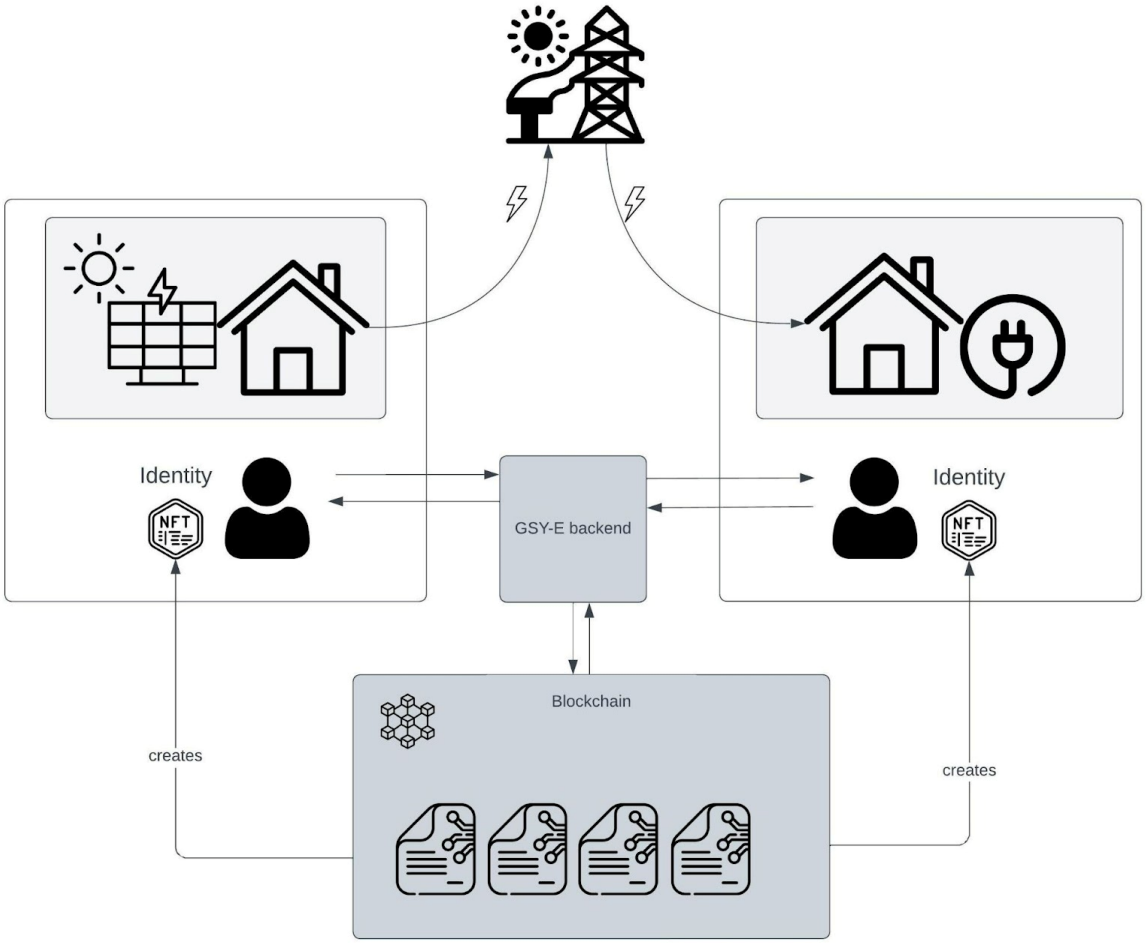
One of the significant advantages of this system is that it eliminates the need for a centralized authority to manage assets and tokens. Instead, the Ethereum blockchain provides a decentralized and trustless system that ensures the safety and security of user assets.

Furthermore, having the same key for both the token and assets enables the interface to detect the identity of the user and associate their assets with that key. This feature provides a convenient way for users to manage their assets and track their ownership and value over time.

3.3 Example

Bob and Alice are two prosumers that intend to exchange money for electricity. Bob owns a solar panel that generates on average 7kwh. His household consumes on average 4kwh. When he has extra electricity, he would like to sell it on the market.

The figure belows shows a simplified schema of this scenario:



Alice has an household where she consumes 2kwh on average. She's interested in purchasing energy from the market.

Both Bob and Alice log into the interface through their meta mask account.
Metamask is connected to the bc4p network with a unique account owned by Bob and Alice respectively.

Before interacting with the bc4p system, bob and Alice create an account through the interface. Their Ethereum public keys are available to the interface and is sent to the bc4p backend together with the user information provided at registration. The backend makes calls

to the SoulBoundToken contract to register the identity of Bob and Alice creating a new token for each one of them.

Now that bob has its identity registered he would like to register his assets in the bc4p system. He first registers his house as an aggregator or “community” in the system then proceeds to register his solar panel within his house by providing all the information about the solar panel including the energy available to sell at any given time and the price he is willing to sell it for. This information is sent to the backend where it is used to deploy Assets smart contracts which submits energy offers to the market when they produce surplus energy.

Alice performs the same registration for the loads present in her house. The backend submints bids for her loads to the maket when it is in need of energy.

One sunny morning Bob was producing 3 kwh extra from the solar panels and Alice was in need of 2kwh of energy as she was running appliances. The Asset contract registers this activity and sends Alices’s bid for 2kwh at 17 cents/kwh and Bob’s offers for 3kwh at 15cents/kwh to the market. The market contract matches Bob’s offer to Alice’s bid and triggers an automatic transaction for the money from Alice to Bob’s account. The final accepted offer is for 17 cents/kwh for 2kwh. The Market smart contract triggers an automatic transaction of 34 cents from Alice’s accounts to Bob’s account.
The remaining 1 kwh in Bob’s offers remains in the market until a new matching bid is found.
Bob and Alice are participants in a peer-to-peer energy trading platform, built on top of the bc4p blockchain network, that enables them to trade electricity in real-time. Bob owns a solar panel that generates 7kWh on average, while his household consumes 4kWh. He wants to sell his surplus electricity on the market, while Alice wants to purchase energy from the market for her household consumption.

To participate in the platform, Bob and Alice log in through their Metamask accounts, which are connected to the bc4p network with their unique accounts. They create accounts on the platform, and their Ethereum public keys are sent to the bc4p backend, along with their user information, to register their identity in the platform. The backend makes calls to the SoulBoundToken contract to create a new token for each of them, which is used to track their transactions on the platform.

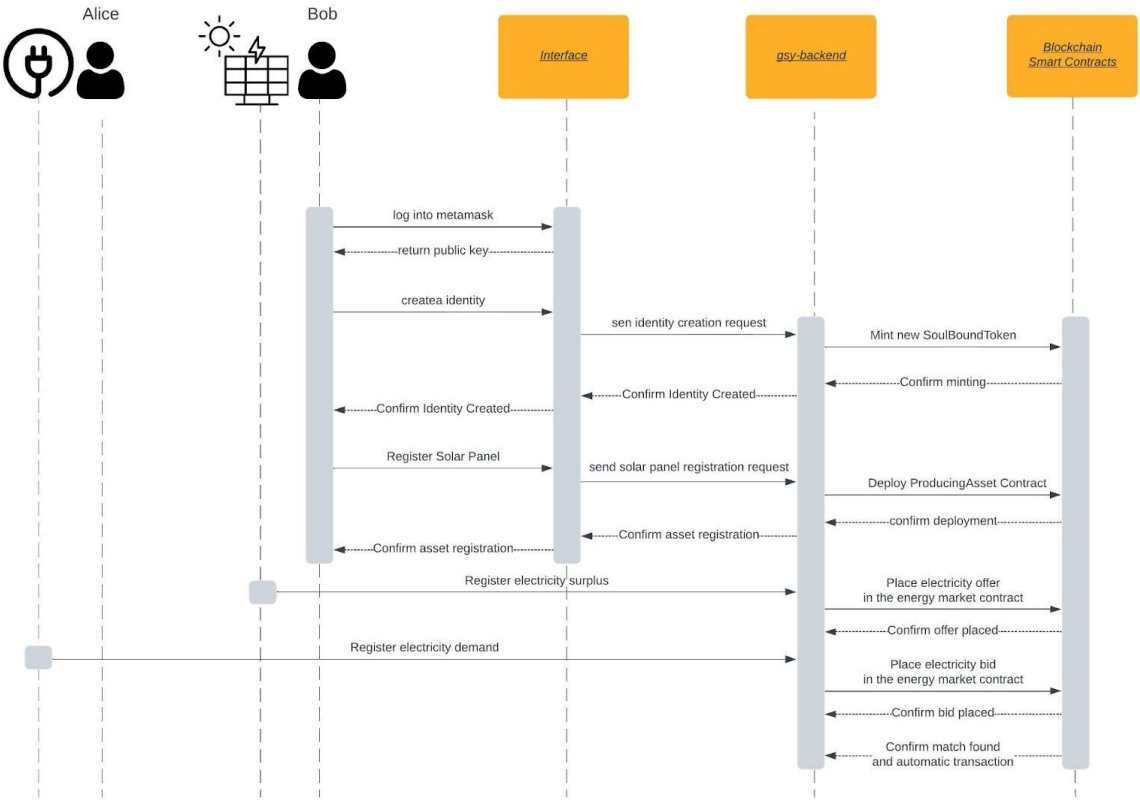
Bob registers his house as an aggregator or “community” in the system and registers his solar panel as an asset within his house. He provides all the relevant information about his solar panel, including the energy available to sell at any given time and the price he is willing to sell it for. This information is sent to the backend, where it is used to deploy the Assets smart contracts. These contracts submit energy offers to the market when Bob's solar panel produces surplus energy.

Alice registers the loads present in her house in a similar fashion, and the backend submits bids for her loads to the market when she needs energy.

When Bob has surplus energy to sell, the Asset contract registers this activity and sends Alice’s bid for 2kWh at 17 cents/kWh and Bob’s offer for 3kWh at 15 cents/kWh to the market. The Market smart contract matches Bob’s offer to Alice’s bid and triggers an automatic

transaction for the money from Alice to Bob's account. The final accepted offer is 17 cents/kWh for 2kWh. The Market smart contract then triggers an automatic transaction of 34 cents from Alice's account to Bob's account. The remaining 1kWh in Bob's offer remains in the market until a new matching bid is found.

The sequence diagram shows the identity registration process for Bob, as the one of Alice is very similar.



3.4 Minting Process

```
// Mapping owner address to token count
mapping(address => uint256) private _balances;
```

With NFT, minting corresponds to creating a new token. In practice, minted tokens are represented as a mapping from ETH addresses to an integer representing the token balance of the address.

Miting is achieved by a simple function that increases the token balance for a specific user, in the case of the ERC721 smart contract:

The soulBoundToken extends the minting functionality by accepting a string that encodes the information of the identity:

```
function safeMint(address to, string memory _image) public onlyOwner {
    require(!owners[to], "Token is aleady owned");
    uint256 tokenId = _tokenIdCounter.current();
    _tokenIdCounter.increment();
    _safeMint(to, tokenId);
    image[to] = _image;
    owners[to] = true;
}
```

3.5 Transferring Tokens

Transfering NFTs between addresses it is a simple matter of increasing the balance of the receiving address and decreasing the balance of the sending address by the specified amount after checking the sender owns the NFT.

In the case of the ERC721 contract:

```
function _transfer(address from, address to, uint256 tokenId) internal virtual {
    require(ERC721.ownerOf(tokenId) == from, "ERC721: transfer from incorrect owner");
    require(to != address(0), "ERC721: transfer to the zero address");

    _beforeTokenTransfer(from, to, tokenId, 1);

    // Check that tokenId was not transferred by `_beforeTokenTransfer` hook
    require(ERC721.ownerOf(tokenId) == from, "ERC721: transfer from incorrect owner");

    // Clear approvals from the previous owner
    delete _tokenApprovals[tokenId];

    unchecked {
        // `_balances[from]` cannot overflow for the same reason as described in `_burn`:
        // `from`'s balance is the number of token held, which is at least one before the current
        // transfer.
        // `_balances[to]` could overflow in the conditions described in `_mint`. That would require
        // all 2**256 token ids to be minted, which in practice is impossible.
        _balances[from] -= 1;
        _balances[to] += 1;
    }
    _owners[tokenId] = to;

    emit Transfer(from, to, tokenId);

    _afterTokenTransfer(from, to, tokenId, 1);
}
```

The SoulBoundToken, by design, requires that the minted token are not transferable though it also implements the transfer function of the ERC721 contract as it extends the contract. To block this functionality, the SoulBoundToken overrides the _beforeTokenTransfer function to block incoming transfer requires

```
function _beforeTokenTransfer(address from, address to, uint256) pure override internal {
    require(from == address(0) || to == address(0), "This a Soulbound token. It cannot be transferred. It can only be burned by the token owner.");
}
```

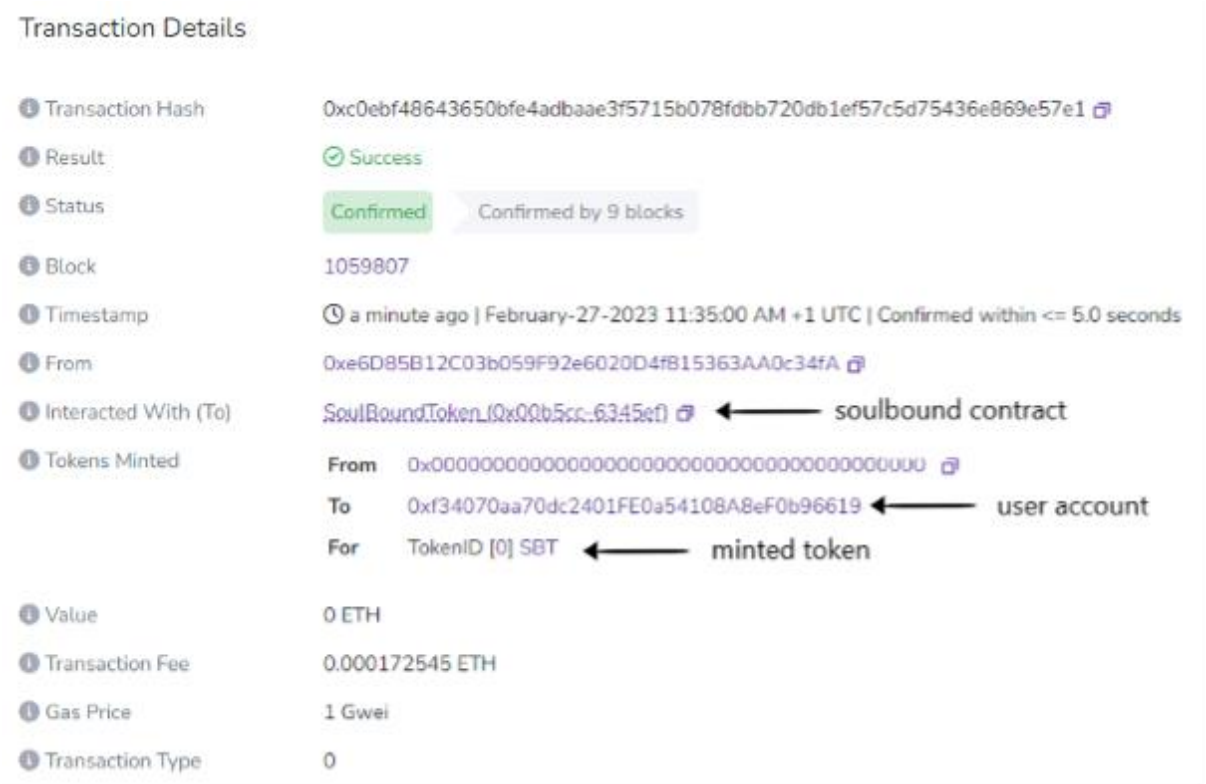
Example: Digital Identity verification for users and assets

In the following example, we will explore an end-to-end example of how digital identities are used within the gsy-e system.

Any user wishing to create an asset needs at least an account in the BC4P blockchain network; accounts consist of a generated pair of private and public keys. Accounts on the blockchain can be created through a wallet provider, such as a meta mask, or they can be created programmatically by the redisConnector. Accounts are ultimately what is mapped to the identity in the SoulboundDNFT contract.

```
def handle_account_registration(event):
    data = json.loads(event["data"])
    new_account = b4p.Accounts.new(data["transaction_id"])
    b4p.SoulboundNFT.safeMint(new_account)
```

When a user obtains a blockchain account, they can submit a request to create an asset in the gsy-e system. The request will generate a redis event for the asset registration, which the redisConnector will handle. The redisConnector will then create a new identity for the account if one does not exist for the blockchain account that made the request.



The identity is created when the redisConnector calls the mint function of the Soulbound NFT contract. The transaction must be signed by the account for which the identity is being

created. At this point, the Soulbound DNFT contract will process the transaction and accept it if there is no existing identity mapped to the account requesting it.

While creating the identity, the user can pass information in the minting function, which will be linked to the identity.

After the identity is created, at any point, the user can retrieve its identity token and the information that is connected to it. The user can also update the information stored in the token or delete the token entirely.

Effectively speaking the generated NFT connects the assets created in GSY to produce and consume energy with the MetaMask accounts of the physical owner of the assets.
Each physical owner can therefore have multiple assets that transact energy in GSY, the soulbound token allows to stored the ownerships of such assets and therefore produce money transactions according to the energy exchanced by the asset of the physical owners.

4. Conclusion and Future Work

This report investigates the digital identities of the user wallets using blockchain technology's implementation in the energy trading use case. Specifically, it seeks to explore how decentralized energy trading platforms utilizing smart contracts can be facilitated by the blockchain to identify the uniqueness of the user and verifier the user, also mapping each asset corresponding to the user with the user profile using NFTs, which helps monetary transactions between producers and consumers without third-party authorization and by simply calling functions in smart contracts. Implementing such systems holds the potential to trade the energy market by providing a secure and transparent means of conducting trade.

Starting from the architecture defined in this deliverable, future works imply the definition of specialized smart contracts to represent future energy production and consumption transactions, thus allowing further mechanisms to profit or optimize energy for prosumers.

5. References

[1] zelfenergieproduceren.nl. Afbouw salderingsregeling uitgesteld tot 2025. URL https://www. zelfenergieproduceren.nl/nieuws/afbouw-salderingsregeling-uitgesteld-tot-2025/.

[2] https://www.blockchain4prosumers.eu/

[3] https://doi.org/10.1016/j.procs.2022.11.238.

(https://www.sciencedirect.com/science/article/pii/S1877050922019482)

[4] https://docs.openzeppelin.com/

[5] Zheng, Gavin & Gao, Longxiang & Huang, Liqun & Guan, Jian. (2021). Decentralized Application (DApp). 10.1007/978-981-15-6218-1_9.

[6] https://eips.ethereum.org/EIPS/eip-20

[7] https://eips.ethereum.org/EIPS/eip-721

[8] https://eips.ethereum.org/EIPS/eip-777

[9] Alsayed Kassem J, Sayeed S, Marco-Gisbert H, Pervez Z, Dahal K. DNS-IdM: A Blockchain Identity Management System to Secure Personal Data Sharing in a Network. *Applied Sciences*. 2019; 9(15):2953. https://doi.org/10.3390/app9152953

[10] Binance Academy. Zero-knowledge proofs. URL https://academy.binance.com/en/glossary/ zero-knowledge-proofs.

[11] Glen Weyl, Puja Ohlhaver, and Vitalik Buterin. Decentralized society: Fiding web3's soul. 2022. URL https://ssrn.com/abstract=4105763.

[12] Boopathi Krishnan. Develop dynamic nfts (dnfts) to expand the boundaries of static nfts, 2022. URL https://morioh.com/p/0c443abc58ec

[13] Chainlink. What is a dynamic nft (dnft)? URL https://chain.link/education-hub/ what-is-dynamic-nft.

[14] https://eips.ethereum.org/EIPS/eip-5484

[15] Sam Daley. What is blockchain technology? how does it work?, 2022. URL https://builtin.com/ blockchain.