Toward a Secure Tokenized Green Credit Management System

No Author Given

No Institute Given

Abstract. Renewable energy registries are centralized systems that track and manage Renewable Energy Certificates (RECs). In this work, we study the Western Renewable Energy Generation Information System (WREGIS), a widely used web-based tracking system in the western United States and Canada, and show that, by requiring a minimum amount of energy production, it introduces intermediaries in the system that prevent small producers to be fairly rewarded. We then propose a blockchain-based system for green credit management that uses a combination of private-public blockchains to remove the intermediaries and allows small producers to directly receive green credit tokens for their energy production and be able to participate in the renewable energy market. The system allows verification of users' claims and issuance of tokens using smart contracts on the private and public blockchains, respectively. We show the correctness and security of the token generation process, and argue that the system provides a high level of privacy for participating users. We also provide descriptions of smart contracts in the two chains and details of our proof-of-concept implementation of the system, and evaluate transaction costs and efficiency of the system.

Keywords: Blockchain and Smart Contract · Renewable Energy · Green Credit Tokenization.

1 Introduction

The renewable energy (RE) market encompasses the production, distribution, and trade of energy generated from sustainable sources such as solar, wind, hydro, geothermal, and biomass. This market has experienced rapid growth in recent years due to increasing concerns about climate change, energy security, and the declining costs of renewable technologies. According to the International Energy Agency (IEA), renewables accounted for nearly 30% of global electricity generation in 2022, with solar and wind energy leading the expansion [20]. Governments worldwide have implemented policies, subsidies, and incentives to accelerate the adoption of clean energy, while corporations and investors are increasingly committing to net-zero goals. Technological advancements, improved energy storage solutions, and growing consumer awareness have further fueled the market's expansion, making renewables a dominant force in the global energy transition [35].

A credit system that rewards renewable energy production provides incentives for individuals and businesses to generate clean energy by issuing renewable energy credits (RECs) or carbon credits. These credits represent proof that a certain amount of electricity was generated from renewable sources and can be traded or sold in environmental markets. For example, in the Renewable Energy Certificate (REC) system

used in the U.S., solar or wind energy producers receive RECs for each megawatt-hour (MWh) of electricity they generate. These RECs can then be sold to companies seeking to offset their carbon footprint, creating a financial incentive for renewable energy production.

In line with *Web3* applications, which emphasize decentralization, transparency, and user participation, small renewable energy producers must be empowered to actively participate in the energy market. By leveraging peer-to-peer (P2P) energy trading, blockchain-based smart contracts, and decentralized energy exchanges, small producers can have greater market access, receive fair compensation, and contribute to energy democratization. This will not only enhance grid resilience, reduce reliance on large utilities, and promote sustainability by incentivizing local clean energy generation. It will incentivize new economic activities through the sharing of resources and exchanges between neighborhoods.

The growing call for sustainable development and renewable energy (RE) generation has attracted an increasing number of households to participate in renewable energy generation using micro-generation technologies [36,40]. The RE rewarding system (REC), however, has not kept up pace and does not provide support for small providers to enter the energy market, nor fairly rewards them for their contribution to sustainability and the green economy. Typically, small producers use their generated energy for their household and "sell" their energy generation surplus to the central grid at a fixed rate determined by their electricity retailer (i.e., the utility service provider). To incentivize renewable energy production, *green credit systems* provide monetary incentives in the form of credits, promoting sustainable and eco-friendly activities [16,4]. Green credit systems, such as Renewable Energy Certificates (RECs) and Carbon Credits (CCs), effectively integrate environmental protection into the financial system, providing attractive opportunities for new markets and financial activities.

In a basic *green credit system*, RE producers report their energy generation data to a *credit-issuing authority*, which verifies their claims and issues credits. Since many credit-issuing agencies impose a minimum threshold on energy generation, small-scale producers rely on *aggregator service* to combine and verify claims and and submits them in a batch, reaching the required minimum. Aggregators, often local electricity retailers or intermediaries (with infrastructure to verify a RE generation claim), receive RECs from credit-issuing authority on behalf of producers. In many cases, including the case of Alberta's micro-generation system [3], the case study considered in this paper, aggregators do not distribute the received RECs to individual producers. Instead, producers benefit from lower energy bills (based on a fixed rate), while aggregators collect green credit rewards from the grid owner (e.g., the local government) and trade the RECs in energy markets, effectively excluding small producers from today's vibrant energy market participation. Credit issuing process may involve other intermediaries. In WREGIS [41], RE data must be submitted through verified entities like Qualified Reporting Entities (QREs), which, like aggregators, are registered with the credit-issuing authority.

Introducing multiple levels of intermediaries, while important to provide verifiability for the claimed generation, creates complexity in the system as well as new opportunities for the misuse of the system. Intermediaries may find incentives to collude with energy producers in incorrect reporting of renewable energy generation data and ultimately, the generation of RECs for invalid claims. In other words, existing ar-

chitecture is a distributed system with many points of trust for correct functioning, while, because of financial benefits, incentivizing rational entities in the system to deviate from their assigned roles.

Blockchain-based credit management. One of the main applications of blockchain in today's information management systems has been the removal of intermediaries, empowering individual nodes to participate in wider exchanges with other nodes. Blockchain-based solutions for energy credit management, including Toucan [39] and Moss [27], aim to decentralize carbon credit management by leveraging Web3 technology to tokenize carbon credits and enable automated peer-to-peer (P2P) trading through smart contracts. However, these systems primarily focus on credit trading and still rely on trusted human verification for renewable energy generation claims and credit issuance. Similarly, systems such as Brooklyn Microgrid [14], which enables peer-to-peer trading of solar energy, and UrbanChain [23], which facilitates energy exchange trading, focus mainly on energy trading rather than fully automating the Renewable Energy Certificate (REC) generation and management process. These systems still depend on intermediaries or centralized components (see Appendix A for more details on these schemes).

Our Work. Our goal is to explore and identify the challenges of modernizing green credit systems through a concrete case study. We consider a web-based renewable energy tracking system that is currently deployed in Canada[1] and uses the Western Renewable Energy Generation Information System (WREGIS) [41] framework. WREGIS is a leading independent renewable energy tracking system that is designed to support the creation, verification, and trading of Renewable Energy Certificates (RECs) across the Western Electricity Coordinating Council (WECC) region, with the goal of ensuring transparency and accountability in renewable energy markets.
(i) Architecture. In Section 3.1, we review the WREGIS workflow and propose a *hybrid blockchain architecture* (in Section 3.2) that combines public and a private blockchains. This hybrid architecture achieves the same goal as WREGIS and ensures: (i) the correct issuance of credits and (ii) user privacy in a blockchain-based green credit management system. The private blockchain, along with its smart contracts and oracles, is used to verify users' energy generation claims. Once verified, these claims are presented to the public blockchain in a verifiable form, where *Green Tokens (GTs)* are issued to the users using a `GToken` smart contract on the public chain. The system effectively reduces reliance on intermediaries by leveraging Oracle smart contracts, which validate users' claims by querying local electricity providers that directly interact with energy producers. By utilizing smart contracts within the private blockchain, the entire verification process for Renewable Energy Certificate (REC) generation is automated, enhancing both transparency and efficiency.
(ii) User privacy. Ensuring that users' energy generation history remains private introduces challenges that ultimately have led to hybrid architecture design to provide an efficient solution. While using cryptography, one can ensure user privacy on public blockchains, but taking into account the high computational cost of zero-knowledge proofs for verifiability and transaction delays, we chose hybrid architecture that allows us to use much more efficient cryptographic primitives to provide user privacy, while providing the ability to expose the required information to the public blockchain in a

[1] Province of Alberta

verifiable way, as required by the token generation. We use BBS+ Signature scheme, [37], a cryptographic scheme that enables efficient and privacy-preserving signing and verification of a set messages. BBS+ allows a signer to generate a single signature on a set of messages, from which the recipient can selectively disclose subsets of signed messages without revealing the entire set. This makes BBS+ particularly useful for privacy-enhancing applications such as anonymous credentials and verifiable credentials in decentralized identity systems. Our system design is detailed in Section 4.
(iii) Analysis. In Section 4.4, we provide a systematic security and privacy analysis of our proposed design. We consider the system entities and their incentives to misuse the system individually or in collusion with others and show that using smart contracts effectively removes the ability of bad actors to result in the incorrect issuance of tokens. We use *unlinkability of a user's transactions* as the main notion of privacy. We define *single-chain unlinkability* , which captures the unlinkability of a user's transaction against a node of a chain, and *cross-chain unlinkability* against an adversary that has nodes in both chains. We argue that our proposed construction satisfies single-chain unlinkability for both chains, as well as cross-chain unlinkability. We also provide a proof-of-concept implementation of our system to show the applicability of the design in practice. We design and implement (in Solidity) a set of two smart contracts for the data verification on the private chain and a smart contract (`GToken`) for token generation on the public blockchain. To estimate the overhead of the cryptographic components of the system in practice, we use a web-based tool [29] to generate BBS+ signature and proof of ownership of the signature. The signature is used in a verifiable credential that we implemented in the W3C standard [25]. This allows us to evaluate the computation and storage cost of the public blockchain for issuing tokens. Our results are presented in Tables 1 and 2 of Section 5.
Organization. Section 2 is preliminaries. Section 3.1 and 3.2 provide WREGIS outline and our blockchain-based design. Section 4 outlines our green token generation scheme, including its security and privacy analysis. Section 5 presents implementation and evaluation results, and Section 6 concludes the paper.

2 Preliminaries

Blockchain and smart contracts. Blockchain is a decentralized, distributed ledger system that securely records transactions across multiple nodes in a network. It ensures data transparency and that the data cannot be altered by any participant without the consensus of the network participants. This work uses two types of blockchains-*public* and *private*. Public blockchains are fully decentralized networks where participants can join and validate transactions, such as Bitcoin [28] and Ethereum [10]. In contrast, private blockchains such as R3 Corda [8], Hyperledger [11] operate within a more controlled environment, with access restricted to specific participants, offering greater control, efficiency, and privacy. Blockchains such as Ethereum also support smart contracts, which are self-executing programs that automate transactions based on predefined rules and allow the development of blockchain-based decentralized applications (dApps). One key blockchain application is tokenization, where assets are represented as digital tokens. The ERC-20 standard, widely used in Ethereum-based tokenization, defines a common set of functions (*totalSupply*, *balanceOf*, *transfer*, *approve*, and *transferFrom*) for new token generation.

BBS+ Signature scheme. The BBS+ signature is a multi-message digital signature protocol that enables signing multiple messages with a single signature while supporting selective disclosure [37,12]. It operates on a pairing-friendly bilinear group where a signer generates a secret-public key pair and signs messages using random generators and cryptographic pairing operations. Verification ensures the signature's validity via bilinear equality checks. Additionally, BBS+ proofs allow proving possession of a valid signature while revealing only a subset of signed messages, maintaining unlinkability and proof of possession. Appendix B provides further details on its main functions with an example.

Verifiable Credential (VC). A *credential* is a digital representation of *claims* (e.g., specific attributes or facts) about a *subject* (e.g., a person or organization), similar to a driver's license or passport. A **verifiable credential** (VC), following World Wide Web Consortium (W3C) standards [25], uses cryptographic proofs (e.g., digital signatures) to confirm the issuer and ensure data integrity. The W3C VC ecosystem [25] has three parties: (i) *issuer*, who creates and issues the VC; (ii) *holder*, who requests, stores, and shares the VC; and (iii) *verifier*, who authenticates the VC by verifying its digital signature. The holder can create a verifiable presentation (VP) to share full or partial VC data securely. Figures 4a and 4b in Appendix C shows the VC issuance and presentation process and its core components: *metadata* (credential details), *attributes* (subject claims), and *proof* (issuer's digital signature on metadata and claims), respectively. A VC is encoded in JSON [6] or JSON-LD [26] formats.

3 A Blockchain-based Approach to WREGIS

We first introduce WREGIS, the *Western Renewable Energy Generation Information System* and then design a hybrid blockchain-based system that achieves the same functionality.

3.1 WREGIS

The *Western Renewable Energy Generation Information System* (WREGIS) [41] is an independent, web-based tracking system designed to issue *renewable energy certificates* (RECs). A REC serves as proof that 1 megawatt-hour (MWh) of electricity was generated from an eligible renewable energy source. Each REC includes information such as the type of renewable fuel used (e.g., solar, wind, water), the month and year the MWh was generated, the facility location, the customer to whom the REC is issued, and the issuer's signature.

WREGIS consists of the following entities: (1) *Users*, which include individual customers (small-scale unit owners), aggregators (managing multiple small units), and organizations (operating large-scale facilities); (2) *Qualified Reporting Entities (QREs)*, third parties tasked with verifying and submitting generation data to WREGIS for certificate creation; (3) *WREGIS Administrator (WA)*, responsible for overseeing system operations, ensuring compliance, and managing user registration; and (4) *Program Administrators (PAs)*, who manage renewable energy programs, certify generating units and meters, and resolve disputes. At a high level, WREGIS operates as follows (see Figure 5 in Appendix D): A user registers with WREGIS and authorizes a Qualified Reporting Entity (QRE) to report generation data. The WREGIS administrator verifies the data's validity and issues REC to the user's account if valid. If the data is invalid, the user needs to submit correct data verified by the program administrator,

with non-compliance leading to the removal of the generating unit from the system. Figure 1 shows the operation of a traditional *micro-generation credit system*, such as the one in Alberta, Canada [3], utilizing WREGIS where the utility service providers offer the aggregator service to micro RE producers (details in Appendix D).

We identify the following key issues: **(i)** *Micro RE generators cannot report data directly to WREGIS and depend on intermediaries like aggregators or QREs*; **(ii)** *The traditional credit system is centralized and not user-centric, limiting fair compensation for small-scale RE generators supplying surplus energy to the grid*; and **(iii)** *Small RE producers cannot directly participate in energy markets for credit exchange as WREGIS is not connected with other credit systems, does not support credit transfers outside its framework, and typically issues credits to aggregators instead of individual generators.* These issues are discussed further in Appendix D with the assumption on WREGIS entities.

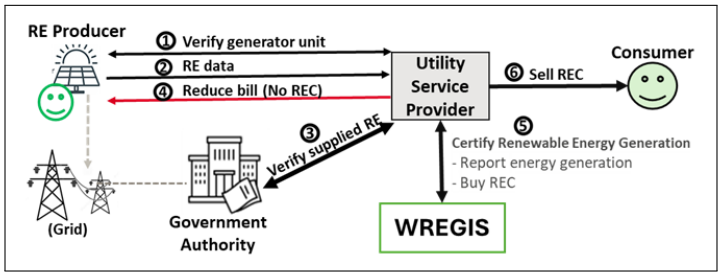


Fig. 1: Workflow of a micro-generation credit system [3].

3.2 Designing a Blockchain-based System for WREGIS

To modernize the WREGIS system by automating renewable energy data verification and certificate generation, we propose a hybrid blockchain-based architecture that combines private and public blockchains for secure green credit management. The *private blockchain* securely stores and verifies renewable energy (RE) generation data, ensuring data integrity and confidentiality, while the *public blockchain* executes the application smart contract for '**green token**' (GT) generation. The system consists of the following main entities:

1. **Users**: The (micro) renewable energy (RE) producers who own small renewable energy generating units (such as rooftop solar panels) and want to receive credit for surplus RE generation.
2. **Registry administrator** (RA): Manages user identities and registration on the registry platform, verifies RE data via smart contracts and deploys an application smart contract for tokenizing RECs.
3. **Governance authority** (GA): Oversees RE regulations, certifies meters, compensates service providers for crediting RE producers, and resolves disputes with on-chain smart contract coordination.
4. **Blockchain**: We use a hybrid model with both private and public blockchains. Users, RA, and GA are nodes on both chains (via client applications).

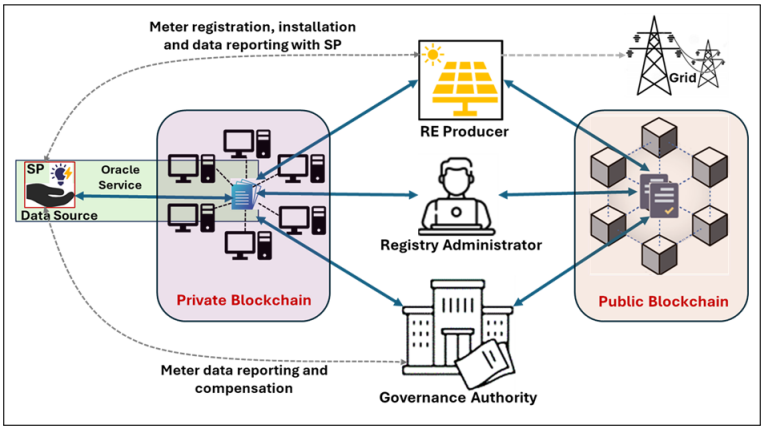


Fig. 2: Overview of Blockchain-based green token system.

a) *Private chain*: Owned by an organization (e.g., the registry) and maintained by its authorized representatives (e.g., RAs). It includes:
   (i) ***Validator nodes*** (RAs): Manage node admission (e.g., user and Oracle nodes) and validate transactions.
   (ii) ***Edge nodes*** (Users): Interact with Validator nodes and authorized smart contracts.
   (iii) ***Oracle nodes*** (part of the Oracle service): Facilitate communication between the private blockchain and external systems.

b) *Public chain*: Receives verified RE data and uses application smart contracts (e.g., `GToken`) for token issuance based on the verified information.

5. **Smart contracts**:
   a) *System contracts*: (Details in Section 4.2.)
      (i) `MetReg` (Meter Registry): Deployed by GA in the private chain to store meter certificates.
      (ii) `DataVer` (Data Verification): Deployed by RA on the private chain to store hashed generation data, verifiable credentials (VCs), and Oracle verification results to validate the authenticity of issued VCs.
   b) *Application contract*:
      (i) `GToken` (Green Token): Deployed by RA on the public chain to verify and issue tokens for valid requests.
6. **Oracle service**: Used by RA to verify user's RE generation claims. It includes an on-chain Oracle contract and off-chain Relay entities (registered as Oracle nodes) that fetch data from trusted external sources (e.g., service providers). Town Crier [42], Muscle [22], Chainlink's Oracle network [7], etc are some examples of existing Oracle services that employ authenticated data feed protocols to ensure secure and reliable data verification.

4 A Green Token Generation Protocol for WREGIS

Trust Assumptions: We make the following trust assumptions:

1. The end users on the private blockchain (e.g., RE producers) are **untrusted**.
2. The registry administrators (RAs) on the private blockchain are **honest-but-rational** and generally follow the protocol as prescribed. However, external incentives, such as financial gain from collusion with a dishonest user, may tempt an RA to deviate from honest protocol participation and issue a verifiable credential (VC) to the user with incorrect RE data.
3. The governance authority (GA) is **trusted** to certify a RE producer's meter (e.g., verify the meter's capacity and link the meter to the owner), enforce policies, and audit other entities in the system.
4. The Oracle system output in private blockchain is **trusted**. Note that in this work, we do not provide a description of any specific Oracle mechanism, as it is outside the scope of this paper. Any existing Oracle system, such as [42,22,7], which guarantees the authenticity of external data, can be adopted.
5. The user's identity and meter information are verified through a trusted authentication process prior to their registration in the private blockchain.
6. Public blockchain observers who do not have access to the private blockchain (i.e., external observers) are **untrusted**.
7. Cryptographic operations such as *BBS+ signature* scheme, verifiable credential (VC), and smart contracts are correctly implemented by the respective entities, and results from smart contract executions are always correct.

Communication Model. We assume that:

1. Communication between the user (RE producer) and the RA happens through a secure channel (e.g., using TLS).
2. Communication between smart contracts (on both public and private chains) and the RA or user takes place over public channels. Furthermore, we assume that a user interacting with smart contracts on the public blockchain can employ multiple pseudonyms derived from a single public key (e.g., to hide their identity).

Threat Model.

1. **Adversaries and Capabilities**: We divide adversaries into the following two categories with their capabilities:
   a) *Private chain adversaries*: The **registered users** in the private blockchain are considered as the potential adversaries in the private network. They have the following capabilities:
      (i) Dishonest users can submit incorrect RE generation data or send the same data multiple times (e.g., send the same RE quantity with a different timestamp) to RA for verification.
      (ii) A dishonest user can forge a verifiable credential by combining arbitrary RE data with a signature from an old VC and request token to public blockchain based on this forged VC.
      (iii) A dishonest user can collude with a registry administrator (RA) to get a VC on incorrect data.
      (iv) Private chain users can observe verification transactions data (submitted to the Oracle contract) on the private chain and token requests (submitted to the `GToken` contract) on the public chain and can attempt to correlate them to compromise RE producers' privacy.

b) *Public chain adversaries*: External observers of the public blockchain who do not have access to the private chain and can see only the data submitted in a token request.

2. **Adversarial goals**:

a) *Incorrect data submission*, *double counting* and *VC forgery* attacks by *dishonest private chain user*: Smart energy meters equipped with Advanced Metering Infrastructure (AMI) technology are typically installed by service providers (SPs) at renewable energy (RE) facilities to enable secure, two-way remote communication between SPs and individual meters [18]. As a result, when reporting generation data to the registry administrator (RA), users need to manually read the data from the meter and submit it to the RA, creating opportunities for dishonest RE producers to submit incorrect generation data for verification.
A malicious user may also attempt a *double-counting attack* by reusing the same generation data multiple times to claim excess credits. This can occur in two ways:
(i) ***Off-chain double-counting***: A dishonest user submits the same RE data to the RA multiple times to claim credits repeatedly.
(ii) ***On-chain double-counting***: A malicious user, after obtaining a valid verifiable credential (VC) from the RA, can attempt to submit multiple token requests using the same underlying signature but generating distinct *BBS+ proofs* by blinding each proof with different randomness, making the proofs unlinkable.
Additionally, a malicious user can *create a forged verifiable credential* by pairing arbitrary renewable energy (RE) generation data with a RA's signature extracted from an old VC and make a token request to fraudulently obtain green tokens.

b) *Collusion fraud* by *the registry administrator*: The RA may collude with a dishonest RE producer if it leads to increased earnings. A colluding RA can attempt to issue a verifiable credential to the dishonest user with incorrect data in different ways. For example, *incorrect data submission to Oracle SC*, submitting multiple verification requests for the same RE data to the Oracle system (i.e., *double-counting attack*), *fraudulent certificate issuance* without performing proper verification, etc., are some of the ways RA can deviate from the original protocol.

c) *Linkability*: In our hybrid architecture, linkability can occur in two forms:
(i) ***Single-chain linkability***: In a blockchain, an adversary may attempt to link multiple transactions belonging to the same user. In the private blockchain, this attack is mitigated through security measures such as encryption and hashing, making transaction linkage infeasible. However, on the public blockchain, adversarial nodes can observe disclosed RE data associated with a token request, potentially linking transactions.
(ii) ***Cross-chain linkability***: An adversary operating nodes on both the private and public blockchains (i.e., a private blockchain adversary) may attempt to correlate private chain data with publicly submitted token request data from the same user, thereby compromising user privacy.

4.1 Security and Privacy Goals

Security goals.

1. Ensure that a green token (GT) is issued only if the RE generation data originates from a certified meter and is reported by its rightful owner.
2. Prevent multiple GT issuances for the same RE data.

Thus, a green token will only be issued to a requester (e.g., a renewable energy producer) if the generation data is authentic, unaltered by adversaries, and accurately reported to the `GToken` smart contract for token generation, a concept known as *Veracity* of data [5]. This protects against adversarial threats, including *incorrect data submission*, *double counting*, *credential forgery*, and *collusion fraud*. The following conditions must be met to achieve our security goals:

(i) *Meter verification*: Ensuring that the meter used for data reporting is authentic (i.e., certified by a trusted certificate authority) and owned by the RE producer reporting the data.

(ii) *Data reading verification*: Validating whether the reported generation data (by both RA and RE producer) accurately reflects the original data.

(iii) *Verifiable data portability across blockchain networks*: Ensuring that verified data can be securely and verifiably transferred from the private blockchain to the public blockchain for token generation.

Privacy goals. We consider the following privacy requirements:

1. (*Cross-chain unlinkability*) Private chain adversaries should not be able to link an RE producer's token request on the public blockchain to a corresponding data verification request on the private blockchain.
2. A public blockchain adversary must not be able to infer an RE producer's $meterID$, facility location (e.g., $siteID$), or any other identifiable attribute by observing a token request.
3. (*Single-chain unlinkability*) Both private and public blockchain adversaries should not be able to correlate whether multiple token requests originate from the same RE producer by observing and analyzing the disclosed data.

For privacy analysis, we consider two scenarios of user interactions:
(i) *Private chain interactions:* A registered RE producer submits RE generation data to the RA, which verifies it via `MetReg` and an Oracle service. Upon validation, the RA issues a verification certificate to the user, which is then used to request token issuance on the public chain.
(ii) *Public chain interactions:* A token requester (e.g., RE producer) submits token issuance requests to the `GToken` contract on the public chain and receives green tokens in return.

4.2 Smart Contracts (SCs) in Token Generation

The details of the contracts are given below (see Appendix E for the abstract of the contracts).

1. **Meter registry (`MetReg`):** Deployed and maintained by the governance authority (GA) in the private blockchain. It stores a meter's digital certificate (signed by the GA) and its status (active/inactive), enabling the registry administrator (RA) to verify meter information upon receiving RE data from a producer. This contract has the following main functions:

a) *registerMeter():* is called by the GA to register a new meter by providing the digital meter *certificate* and sets the initial status of the meter, such as *active* or *inactive*.
b) *verifyMeter():* is called by RA to initiate meter verification. If the meter has an active status, then the contract verifies the meter's certificate using GA's public key. If successful, it returns *true*; otherwise, $false$.
c) *updateMeterStatus():* allows GA to update the status of a meter.
d) *verifyGASignature():* a private helper function to verify GA's signature.

2. **Data verification (`DataVer`):** Deployed by the RA on the private blockchain, it serves as a directory to store the verification requests. The main functions of this SC are:
   a) *submitRECommitment()*: is called by a user to submit a hashed commitment of generation data for verification.
   b) *submitOracleVerification()*: is called by the Oracle SC to submit verification output after external verification.
   c) *storeVCCommitment()*: is called by RA to store the hashed verifiable credential (VC) after successful verification.
   d) *matchVerificationRequest()*: is called internally by the `storeVCCommitment()` function to verify if the user's commitment matches the Oracle's output for which the VC was issued. If the match is valid, it returns *true*; otherwise, it returns *false*. If *false* is returned (e.g., the user's commitment is not found or the Oracle's output is invalid), the VC is discarded as incorrect, and the Government Authority (GA) is notified (by emitting a dispute message) to resolve the dispute.
3. **Green token generation (`GToken`):** Deployed by RA in the public blockchain and responsible for verifying token requests and, if valid, issues ERC-20 complaint green token(s) to the requester. The key functions are:
   a) *requestGToken()*: It allows a user to submit a valid BBS+ proof to request green tokens (GTs). It verifies the proof, and if valid, it calls the `issueToken()` function internally for token issuance. It calls the `verifyRequestCommitment()` function internally to check for double reporting of the same data.
   b) *verifyBBSProof()*: It verifies the BBS+ proof to ensure that the token requester has valid data (e.g., valid signature from RA on verified RE data). It returns *true* if the proof is valid, *false* otherwise.
   c) *issueToken()*: After successful proof verification, this function is called inside `requestToken()` to issue new green tokens to the requester's address. It uses the relevant ERC-20 function(s) for this token issuance.

4.3 User interactions in Green Token Generation

There are three stages in our token generation scheme:

1. **Registration:**
   a) *User registration*: RA verifies each user's identity through a trusted identity verification process (e.g., verifying a government-issued ID) and issues a login credential to interact with the private blockchain.
   b) *Meter registration*: The GA verifies and certifies electric meters through in-person inspections and creates a certificate $Cert^{GA}_{meterID}$ containing hashed

meter, owner, and site IDs, meter attributes, a timestamp, and GA's signature ($Cert^{GA}_{meterID}$ = *{H(meterID),H(ownerID),H(siteID),meterAttributes, verificationDate,*σ_{GA}*}*). GA registers the meter as "*active*" by calling the *registerMeter*() function in the `MetReg` SC.

2. **Data reporting and verification in private blockchain:**
   a) *Data reporting to RA*: Registered users (RE producers) send (*generationData*, σ_{User}) to RA for verification through secure off-chain channel and publish H(*generationData*) on the private chain via the `DataVer` SC as a commitment. Here, *generationData = {ownerID, meterID, siteID, timestamp (start date, end date), REType, REQuantity}* and $\sigma_{User} = Sign_{User_{SK}}$(*generationData*).
   b) *Data verification using Oracle*:
      (i) RA verifies the meter's validity using the *verifyMeter()* function in the `MetReg` contract. If valid, RA encrypts *generationData* with the service provider's (i.e., external data source) public key PK_{SP}, gets $c_{\mathcal{GD}}$ = `Enc`(PK_{SP}, *generationData*) and sends $c_{\mathcal{GD}}$ to Oracle SC for external verification. The Oracle SC first checks if $c_{\mathcal{GD}}$ already exists in its storage. If it does, reject the verification request identifying it as a "*Duplicate Request*". Otherwise, forwards $c_{\mathcal{GD}}$ to SP.
      (ii) The SP decrypts $c_{\mathcal{GD}}$, validates *generationData* and returns ($True$, H(*generationData*), σ_{SP}) to `DataVer` SC via the Oracle contract, where σ_{SP} is the digital signature of SP on the hashed *generationData*. Otherwise, SP returns $False$.
      (iii) If valid, RA issues $VC_{ownerID} = \{metadata, generationData, \sigma_{RA}\}$ to the user, where *metadata* includes VC details like type, issue date, and expiration date, while σ_{RA} is RA's BBS+ signature on *metadata* and *generationData*.
      (iv) RA then publishes ($H(VC_{ownerID})$, H(*generationData*)) to `DataVer` SC for each VC issued against a verified *generationData*.

3. **Token request on public blockchain by user (VC holder)**:
   a) User first verifies RA's signature σ_{RA} on VC and sends a *token_request* (=$\{$*proof*, $M_{Disclosed}$, *public parameters*$\}$) to `GToken` SC, where *proof* is *BBS+ proof* on $VC_{ownerID}$ contents, $M_{Disclosed}$ is the revealed data subset and *public parameters* are for σ_{RA} and *proof* verifications.
   b) The `GToken` SC: (i) Computes $H(M_{Disclosed})$ and verifies whether this hash already exists in its storage. If not, it stores the hash as a commitment to the token request; otherwise, it rejects *token_request*. (ii) Checks the signature and the *BBS+ proof*. If valid, it issues green token(s) to the user. Otherwise, it rejects the *token_request*.

Figure 7 in Appendix F shows the flowchart of the above interactions.

4.4 Security and Privacy Analysis

Security Analysis. The following statement holds for our green token generation protocol:

Statement: *The green token generation protocol ensures secure token issuance by guaranteeing that only authentic, verifiable, and tamper-proof renewable energy data is used, effectively preventing incorrect data submission, signature forgery, collusion fraud, and double counting.*

Required security conditions (mentioned in Section 4.1) are satisfied as below:

1. **Verified meter**: Meter verification occurs on the private chain via the `MetReg` SC. A meter is valid if (a) certified by the GA and (b) operated by its rightful owner. The RA verifies the GA's signature (σ_{GA}) using the GA's public key, ensuring data originates from a certified meter and its owner.
2. **Verified data reading**: RE generation data accuracy is ensured through (i) meter verification (condition (1)) and (ii) Oracle-based cross-verification against trusted external sources. This prevents:
   a) *Incorrect data submission* by dishonest user or colluding RA: Oracle verification fails for any incorrect data submission
   b) *Off-chain double counting* by dishonest user: The `DataVer` SC's *submitRECommitment()* function ensures unique commitments, rejecting duplicates.
   c) *Double counting* by a colluding RA: The Oracle SC rejects any duplicate encrypted data.
   d) *Fraudulent certificate issuance* by colluding RA: The `DataVer` SC's *matchVerificationRequest()* function ensures VCs are only issued for verified data.
3. **Secure data transfer across chains**: The RA signs validated generation data using the BBS+ signature scheme, included in a verifiable credential (VC). Users generate a BBS+ proof from the VC and submit it to the `GToken` SC with a *token_request* containing disclosed RE data ($M_{Disclosed}$) and public parameters. This prevents:
   a) *Credential forgery and arbitrary data submission* during *token_request*: BBS+ signature scheme ensures that adversaries cannot forge VCs or link arbitrary data to RA signatures (e.g., by linking arbitrary renewable energy (RE) generation data with a previously extracted RA signature from an old VC), causing proof verification to fail and token requests to be rejected.
   b) *On-chain double counting* by malicious RE producer: The `GToken` SC's *verifyRequestCommitment()* function checks for duplicate $M_{Disclosed}$ hashes and rejects an on-chain *token_request* if a match is found.

Privacy Analysis. For privacy analysis of the proposed green token generation process, we examine the views of users and other entities in the system to determine what information can be inferred. Tables 3 and 4 in Appendix G detail the visible information during private and public blockchain interactions (as defined in Section 4.1). Our protocol ensures privacy by combining verifiable credentials (VCs), BBS+ signatures for selective disclosure, and a hybrid blockchain model, fulfilling the privacy requirements outlined in Section 4.1.

(i) **Privacy goal (1)** is ensured because:
   a) *For meter verification*, the RA checks the meter's certification details and validates GA's signature (σ_{GA}) by querying the `MetReg` smart contract using the hashed $meterID$, hiding the $meterID$ from other users.
   b) *During data verification*, RA submits encrypted RE data $c_{\mathcal{GD}}$ to the Oracle system, which is decrypted off-chain by the trusted external data source, and only a binary verification result (valid/invalid) is posted on-chain. The RA then sent a BBS+ signed VC to the producer off-chain.

   Thus, RE producer identity and *generationData* remain hidden during both verifications, and observers cannot link private chain requests to public chain token requests, ensuring *cross-chain unlinkability*

(ii) **Privacy goal (2)** is guaranteed because the *token_request* includes a *BBS+ proof* derived from a BBS+ signed VC and a subset of *generationData*. The `GToken` SC verifies the proof, confirming that the user possesses a valid BBS+ signature from the RA and the disclosed subset of RE data is signed as part of that signature. The *selective disclosure* property of BBS+ allows hiding sensitive data (e.g., $meterID$, $siteID$, $ownerID$) while revealing only necessary details (e.g., $timestamp, REType, REQuantity$), preventing public chain observers from linking the token request to a specific meter, location, or producer.

(iii) **Privacy goal (3)** is satisfied by the use of BBS+ signature scheme. Each BBS+ proof in a *token_request* is unique due to varying *generationData* (e.g., different *REQuantity* values at different *timestamp*). Additionally, public chain users can use multiple pseudonyms from a single public key, preventing public chain observers from linking multiple requests to the same RE producer, ensuring *single-chain unlinkability.*

5 Implementation

We provide a proof-of-concept implementation to estimate costs and evaluate the practicality of the proposed token generation process by measuring the *computational time and storage overhead of cryptographic operations* and *gas costs* for executing smart contract functions on the public blockchain. The smart contracts are developed in Solidity using the Remix IDE [19], with Ganache [15] employed for local Ethereum blockchain simulation. A web-based tool [29] is used to generate BBS+ signatures and proofs. Verifiable credentials (VCs) are implemented in Python, following the W3C data model [25] that takes renewable energy (RE) attributes and a BBS+ signature as input and outputs a VC in JSON format. Performance evaluation is conducted on a Windows 11 machine with an Intel(R) Core(TM) i7 CPU @ 3.60 GHz and 8 GB RAM.

***`GToken`** contract implementation challenge and our choice of solution.* The `GToken` contract integrates BBS+ proof verification and ERC-20 token generation. Since Solidity lacks native support for pairing-friendly elliptic curves like BN254 or BLS12-381, implementing an exact BBS+ proof verifier on-chain is challenging. To address this, we use OpenZeppelin's bn256Pairing library for ZKP verification in Solidity, enabling gas cost estimation for proof verification on the public blockchain. See Appendix H for system setup details and VC generation pseudocode.

Evaluation and Performance Analysis. To assess the efficiency of our implementation, we implemented a sample renewable energy (RE) scenario in which a RE producer has a dataset $M = \{$"$did : example : owner789$", "$meter - 56789$", "$site - 34567$", "$Solar$", "$100kWh$", "$2025 - 02 - 15T12 : 00 : 00Z$"$\}$. We measured the computational cost (execution time in milliseconds) and storage overhead (data size in bytes) for cryptographic mechanisms like verifiable credentials (VCs), BBS+ signatures, and zero-knowledge proofs (BBS+ proofs). The results are summarized in Table 1, where- (i) BBS+ signature computational time is excluded (denoted as *NA*) as it uses a web-based implementation [29], (ii) the (*) symbol next to BBS+ proof indicates that the proof size varies based on the number of undisclosed (hidden) messages, and (iii) Oracle service costs are excluded since it is not implemented, but services

such as [42,22,7] can be integrated, with costs varying based on the Oracle service provider.

Table 1: Cost of cryptographic operations.

Component	Computing Entity	Size (bytes)	Time (ms)
BBS+ Signature	RA	160	*NA*
Verifiable Credential	RA	1260	0.25
*BBS+ Proof**	RE Producer	384	*NA*

Table 2: Token request verification and token issuance cost (in gas) in the public chain.

Operation	Gas Cost
ZKP Verification	160,060 gas
Minting GT	67,200 gas
Total cost (estimated)	227,260 gas

After data verification, the producer receives a VC from the RA. When submitting a token request to the `GToken` contract, the producer discloses $M_{Disclosed} = \{\text{“}Solar\text{”}, \text{“}100kWh\text{”}, \text{“}2025-02-15T12:00:00Z\text{”}\}$ while keeping the remaining attributes $M_{Hidden} = \{\text{“}did:example:owner789\text{”}, \text{“}meter-56789\text{”}, \text{“}site-34567\text{”}\}$ private. We measured the gas cost of ZKP verification and token issuance (using OpenZeppelin's ERC-20 library) on the Ethereum test network. Table 2 shows the gas costs for these operations, totaling approximately 0.63% of Ethereum's block capacity[2].

The results in Tables 1 and 2 show that our token generation approach achieves a balance between security and efficiency, with reasonable computational overhead for RE data verification, and the gas cost for on-chain token request verification and issuance are not high. These results validate the practicality of our model, making it adaptable to various EVM-based blockchain ecosystems.

6 Concluding remarks

Green credit is an important tool to incentivize users towards renewable energy generation and, ultimately, a sustainable environment. We pointed out the shortcomings of existing green credit systems that effectively exclude small producers from energy credit systems and the financial market that is built around that, and proposed a blockchain-based system that is a step towards democratization of the energy market by empowering small producers to receive credit for their renewable energy production. Interesting open questions are formal security and privacy analysis of our protocol, and designing credit exchange systems for different types of green credit.

References

1. Puro earth, `https://puro.earth/`, accessed on June 14, 2024
2. Agency, S.E.P.: Renewable energy tracking systems, `https://www.epa.gov/green-power-markets/renewable-energy-tracking-systems`, accessed on June 14, 2024
3. Government of Alberta, C.: Micro-generation, `https://www.alberta.ca/micro-generation`, accessed on February, 2025
4. BanQu: India's green credit programme: What it is how to prepare, `https://www.banqu.co/blog/indias-green-credit-programme-what-it-is-how-to-prepare?`, accessed on February, 2025
5. Berti-Equille, L., Borge-Holthoefer, J.: Veracity of Data. Springer Nature (2022)

[2] Based on block gas limit from etherscan.io/chart/gaslimit on March 01, 2025.

6. Bray, T.: The JavaScript Object Notation (JSON) Data Interchange Format. RFC 8259 (Dec 2017). https://doi.org/10.17487/RFC8259, https://www.rfc-editor.org/info/rfc8259
7. Breidenbach, L., Cachin, C., Chan, B., Coventry, A., Ellis, S., Juels, A., Koushanfar, F., Miller, A., Magauran, B., Moroz, D., et al.: Chainlink 2.0: Next steps in the evolution of decentralized oracle networks. Chainlink Labs **1**, 1–136 (2021)
8. Brown, R.G., Carlyle, J., Grigg, I., Hearn, M.: Corda: an introduction. R3 CEV, August **1**(15), 14 (2016)
9. Bureau, C.: Wepower, https://coinbureau.com/review/wepower-wpr/, accessed on February, 2025
10. Buterin, V., et al.: A next-generation smart contract and decentralized application platform. white paper (2014)
11. Cachin, C., et al.: Architecture of the hyperledger blockchain fabric. In: Workshop on distributed cryptocurrencies and consensus ledgers. vol. 310, pp. 1–4. Chicago, IL (2016)
12. Camenisch, J., Drijvers, M., Lehmann, A.: Anonymous attestation using the strong diffie hellman assumption revisited. In: Trust and Trustworthy Computing: 9th International Conference, TRUST 2016, Vienna, Austria, August 29-30, 2016, Proceedings 9. pp. 1–20. Springer (2016)
13. of Canada, G.: Canada's greenhouse gas offset credit system, https://www.canada.ca/en/environment-climate-change/services/climate-change/pricing-pollution-how-it-will-work/output-based-pricing-system/federal-greenhouse-gas-offset-system.html, accessed on June 14, 2024
14. Energy, B.: Brooklyn microgrid, https://www.brooklyn.energy/, accessed on February, 2025
15. Ganache: (2019), https://www.trufflesuite.com/ganache
16. He, L., Zhang, L., Zhong, Z., Wang, D., Wang, F.: Green credit, renewable energy investment and green economy development: Empirical analysis based on 150 listed companies of china. Journal of cleaner production **208**, 363–372 (2019)
17. (IATA), I.E.T.A.: Carbon credit 101, https://ieta.b-cdn.net/wp-content/uploads/2023/09/IETA_101_CarbonCredits_Sept2023.pdf, accessed on June 16, 2024
18. IBM: What is advanced metering infrastructure (ami), https://www.ibm.com/think/topics/advanced-metering-infrastructure, accessed on February, 2025
19. solidity IDE, R.: (2019), https://remix.ethereum.org
20. (IEA), I.E.A.: Renewables 2023: Analysis and forecasts to 2028 (2024), https://www.iea.org/reports/renewables-2023
21. International, W.: American carbon registry, https://acrcarbon.org/acr-registry/, accessed on June 14, 2024
22. van der Laan, B., Ersoy, O., Erkin, Z.: Muscle: Authenticated external data retrieval from multiple sources for smart contracts. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. pp. 382–391 (2019)
23. Ltd, U.: Urbanchain, https://www.urbanchain.co.uk/, accessed on February, 2025
24. M-RETS: M-ret, https://www.mrets.org/about/mission-vision-values/, accessed on June 14, 2024
25. Manu Sporny, D.L., Chadwick, D.: Verifiable credentials data model v1.1 (2022), https://identity.foundation/bbs-signature/draft-irtf-cfrg-bbs-signatures.html#name-generators
26. Manu Sporny, Dave Longley, G.K.M.L.P.A.C., Lindström, N.: Json-ld 1.1 (2020), https://www.w3.org/TR/json-ld11/
27. MOSS: Moss whitepaper, https://v.fastcdn.co/u/f3b4407f/54475626-0-Moss-white-paper-eng.pdf, accessed on June 14, 2024
28. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
29. Networking, G.: Bbs signature demo, https://www.grotto-networking.com/BBSDemo/

30. OpenZeppelin: Erc-20, https://github.com/OpenZeppelin/openzeppelin-contracts/tree/master/contracts/token/ERC20
31. Patel, S.: A guide to renewable energy credits (recs)., https://esg.conservice.com/guide-to-renewable-energy-credits/, accessed on June 14, 2024
32. PONTON: Enerchain, https://www.ponton.de/enerchain, accessed on February, 2025
33. Powerledger: Powerledger, https://powerledger.io/, accessed on February, 2025
34. Registry, V.: Verra, https://registry.verra.org/?_gl=1*1e9wpvm*_ga*MTg0MDg5MzE1Ni4xNzEzMjk2NDI1*_ga_2VGK901B6P*MTcxMzI5NjQyNS4xLjEuMTcxMzI5NjU5NS4wLjAuMA, accessed on June 14, 2024
35. REN21: Renewables 2023 global status report (2023), https://www.ren21.net/gsr-2023/
36. Scarpa, R., Willis, K.: Willingness-to-pay for renewable energy: Primary and discretionary choice of british households' for micro-generation technologies. Energy Economics **32**(1), 129–136 (2010)
37. T. Looker, V. Kalos, A.W., Lodder, M.: The bbs signature scheme (2024), https://identity.foundation/bbs-signature/draft-irtf-cfrg-bbs-signatures.html#name-generators
38. Toucan: Carbon-credit bridging approval process in toucan., https://docs.toucan.earth/toucan/carbon-bridge/puro-carbon-bridge/await-approval, accessed on June 14, 2024
39. Toucan: Toucan, https://docs.toucan.earth/, accessed on June 14, 2024
40. Willis, K., Scarpa, R., Gilroy, R., Hamza, N.: Renewable energy adoption in an ageing population: Heterogeneity in preferences for micro-generation technology adoption. Energy Policy **39**(10), 6021–6029 (2011)
41. WREGIS: Western renewable energy generation information system (wregis) - operating rules, https://www.wecc.org/Administrative/WREGIS%20Operating%20Rules%20October%202022%20Final.pdf, accessed on June 14, 2024
42. Zhang, F., Cecchetti, E., Croman, K., Juels, A., Shi, E.: Town crier: An authenticated data feed for smart contracts. In: Proceedings of the 2016 aCM sIGSAC conference on computer and communications security. pp. 270–282 (2016)

A Related Work

Several greenhouse gas emission tracking and credit systems exist to support sustainability, both blockchain-based and non-blockchain-based. Non-blockchain systems, such as web-based platforms for carbon offset credits [13,34,1,21] and renewable energy tracking [24,2,31], rely on trusted intermediaries (e.g., aggregators) or centralized entities (e.g., local energy providers) for data reporting and verification. This limits small-scale RE producers' ability to participate directly in energy trading.

Blockchain technology and smart contracts have gained significant attention in recent years for their inherent characteristics, such as transparency, decentralization, and immutability in various domains, including renewable energy and sustainability. Several blockchain-based systems are being implemented for managing renewable energy certificates (RECs), and enabling peer-to-peer (P2P) energy trading such as [33,9,23,32,14]. [32] and [33] facilitates P2P energy trading and REC management using blockchain tokens, allowing consumers and producers to trade energy directly. [9] is another blockchain-based platform that allows renewable energy producers to tokenize their energy production and sell it directly to consumers. Brooklyn Microgrid [14], a project of LO3 Energy, is another example system that enables residents

to trade solar energy within their community using blockchain technology. UrbanChain [23] is another peer-to-peer energy exchange services in the United Kingdom (UK) that enables consumers and producers to trade renewable energy directly. While innovative, these platforms- do not fully address the challenges of automated, tamper-proof REC verification and issuance, primarily focus on energy trading rather than REC management, or often rely on intermediaries or centralized components.
Blockchain has also been applied to sustainability and carbon credit[3] management [39,27]. These systems aim to develop a decentralized solution and address the issues in traditional tracking systems such as double-counting and incorrect data submission. These systems use Web3 technology to tokenizecarbon credits and enable automated P2P trading[4] through smart contracts. However, despite being a leading blockchain-based solution for carbon credit tokenization, Toucan [39] remains centralized and human-dependent [38], where each carbon credit brought on-chain as a TCO2 token (an ERC-20 standard token) undergoes a verification process carried out by a Toucan verifier, who must be a trusted member of the Toucan community.

B The BBS+ Signature Scheme

It is a multi-message digital signature protocol that allows a signer to sign multiple messages while producing a single digital signature [37,12]. The main functions of this scheme (as defined in [37]) are explained below following the notation of [12]:

1. **Key generation:** Let $(\mathbb{G}_1, \mathbb{G}_2)$ be a bilinear group pair of some prime order p, and g_1, g_2 are two base points on $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively (generated using a *setup* algorithm). The key generation function randomly generates $x \leftarrow \mathbb{Z}_p$ as the *secret key* and computes $w \leftarrow g_2^x$ as the corresponding *public key*.
2. **Signature generation**: The *Sign* function takes the secret key x, public key w and the set of messages $M = \{m_1, m_2, m_3, \ldots, m_L\}$ as input and generates a signature σ. The signature generation process involves the following steps: (i) generate a random generator for each message $\{h_1, h_2, \ldots, h_L\} \leftarrow \mathbb{G}_1$, (ii) generate a random number $\epsilon \leftarrow \mathbb{Z}_p$, (iii) compute $A = (g_1 \prod_{i=1}^{L} h_i^{m_i})^{\frac{1}{\epsilon+x}}$ and (iv) output the signature $\sigma = (A, \epsilon)$.
3. **Signature verification**: The *Verify* function checks the validity of the signature against a public key. Given the public parameters $(g_1, g_2, h_1, \ldots, h_L, w)$, messages $\{m_1, m_2, m_3, \ldots, m_L\}$ and signature $\sigma = (A, \epsilon)$, the signature verification is done by checking the equality $e(A, wg_2^{\epsilon}) = e(g_1 \prod_{i=1}^{L} h_i^{m_i}, g_2)$. Here, e is a bilinear map that satisfies the following properties [12]: bilinearity (e.g., $e(g_1^x, g_2^x) = e(g_1, g_2)^{xy}$), non-degeneracy (i.e., $e(g_1, g_2) \neq 1$) and efficiency.
4. **Proof generation**: The *ProofGen* operation generates a *BBS+ proof*, which is a zero-knowledge proof-of-knowledge of a BBS+ signature with a disclosed subset of the signed messages. It blinds the signature and undisclosed messages with random scalars, with a strict requirement of maintaining the (signed) message order and the indexes of disclosed messages.

[3] Carbon credits are tradable certificates representing a reduction, removal or avoidance of one metric ton of greenhouse gas emissions from the atmosphere [17].
[4] Peer-to-peer (P2P) trading involves the direct exchange of surplus electricity between two parties on a connected grid.

5. **Proof verification**: The *ProofVerify* function validates a *BBS+ proof* (the blinded values), given the Signer's public key, and the disclosed messages and their generators. This verification ensures- (i) the prover has possession of a valid signature and (ii) the disclosed messages are signed as part of the original signature.

The BBS+ signature scheme [37] has three key properties:

(i) *Selective disclosure*: Allows a prover who has the messages, the signature, and group generators used for signing the messages to create a proof by choosing a subset of messages to disclose, while keeping the remaining messages and the signature hidden.
(ii) *Unlinkable proofs*: Proof created by the prover are zero-knowledge proofs of knowledge of the signature. It ensures that multiple proofs derived from the same signature cannot be linked.
(iii) *Proof of possession*: Verifies that the prover holds a valid signature and its signed messages, without revealing the signature itself.

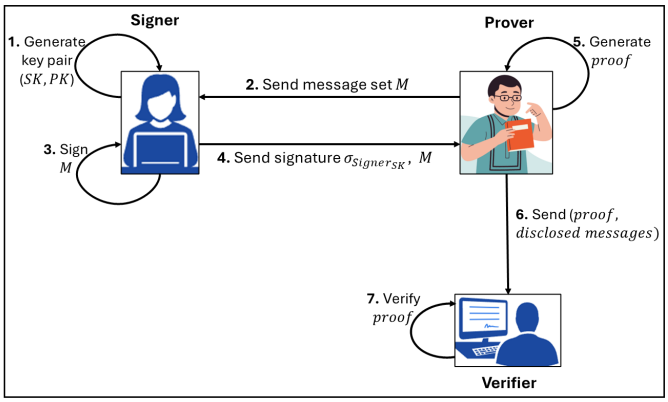


Fig. 3: BBS+ signature scheme [37], where SK and PK are Signer's secret key and public key, respectively, M denotes the message set and $proof$ is a BBS+ proof created using the proof generation function (`ProofGen`) in [37].

An Example of BBS+ Signature Scheme Consider a scenario where a user (Prover) has three messages, $M = \{m_1, m_2, m_3\}$, which were sent to a Signer. The Signer generates a BBS+ signature, σ, on the message set M and returns it to the user along with the messages. If the user wants to reveal m_2 to a Verifier while keeping m_1 and m_3 private, the BBS+ signature scheme [37,12] enables the user to selectively disclose specific signed messages. Using this scheme, the user can create a zero-knowledge proof that shows possession of a valid signature σ, while revealing only m_2 and keeping the remaining messages (m_1, m_3) and the signature σ hidden from the Verifier. The steps are given below:

(***Signature generation by Signer and verification by Prover***)

1. *Key generation*: Signer generates *secret key* x and *public key* $w = g_2^x$.
2. Prover sends messages $M = \{m_1, m_2, m_3\}$ to Signer.

3. Signer:
   (a) generates message generators $\{h_1, h_2, h_3\}$
   (b) picks random numbers ϵ
   (c) computes $A = (g_1 h_1^{m_1} h_2^{m_2} h_3^{m_3})^{\frac{1}{\epsilon+x}} = B^{\frac{1}{\epsilon+x}}$, where $B = (g_1 h_1^{m_1} h_2^{m_2} h_3^{m_3})$
   (d) outputs signature, $\sigma = (A, \epsilon)$
4. Signer sends $(\sigma, M, \{h_1, h_2, h_3\})$ to Prover

(***Proof generation by Prover and Proof verification by Verifier***)
(Proof generation) Suppose the Prover discloses the message $\{m_2\}$ and keeps $\{m_1, m_3\}$ hidden to a Verifier. Then the Prover creates a signature proof of knowledge (SPK) where certain messages are disclosed and others remain hidden as follows:

1. generate random numbers r_1, r_2
2. compute $B = (g_1 h_1^{m_1} h_2^{m_2} h_3^{m_3})$
3. (to blind A) compute $\acute{A} = A^{r_1 r_2}$ **[eq (i)]**
4. (to blind B) compute $D = B^{r_2}$ **[eq (ii)]**
5. compute $\acute{B} = D^{r_1} \acute{A}^{-\epsilon}$ $(= \pi_1)$ **[eq (iii)]**
6. compute $g_1 h_2^{m_2} = D^{r_2^{-1}} h_1^{-m_1} h_3^{-m_3}$ $(= \pi_2)$ **[eq (iv)]**
7. compute $\pi = SPK\{\left(\epsilon, r_1, r_2^{-1}, m_1, m_3\right) : \pi_1 \wedge \pi_2\}$. π denotes a proof of knowledge of integers $\left(\epsilon, r_1, r_2^{-1}, m_1, m_3\right)$ such that π_1 and π_2 holds.

So, the ***proof*** is: $(\acute{A}, \acute{B}, D, \pi)$ that the Prover sends to the Verifier along with the disclosed message $\{m_2\}$.
[At this point: private values are: $(\ x, \epsilon, r_1, r_2^{-1}, \{m_1, m_3\}, A, B$, and the public values are: $(g_1, g_2, h_1, h_2, h_3, w), \left(\acute{A}, \acute{B}, D, \pi\right), m_2\ ]$

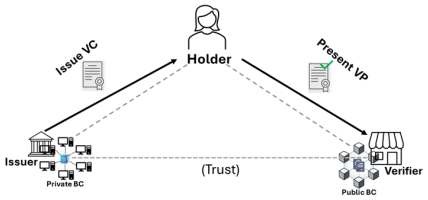
1. (Proof verification) Upon receiving the proof the Verifier checks the following (using the public parameters):
   (a) ***Verify the signature*** σ is valid by checking $e\left(\acute{A}, w\right) = e\left(\acute{B}, g_2\right)$
   (b) **Verify that the disclosed message $\{m_2\}$ is signed as part of the signature** σ. (i.e., Validate π by checking whether equations (iii) and (iv) holds or not)

C W3C Verifiable Credential

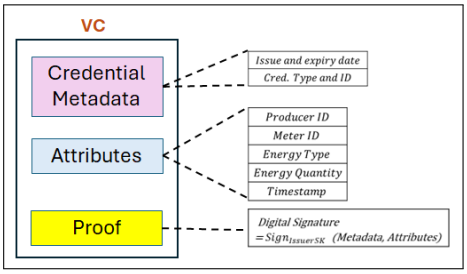
The basic components of a verifiable credential (VC) data model and its lifecycle (as defined by W3C in [25]) are shown in Figure 4a. In Figure 4b, the *metadata* provides information about the credential, *attributes* represent claims about the subject, and *proof* is the cryptographic evidence, ensuring integrity through the issuer's digital signature on both metadata and attributes.

D Issues in WREGIS

Figure 5 shows how WREGIS works and Figure 1 in Section 3.1 illustrates an example scenario of how a traditional micro-generation credit system works utilizing the tracking system, showing the flow of interactions among different participants -



(a) VC issuance and presentation.



(b) Basic VC components with sample values.

Fig. 4: W3C Verifiable Credential (VC) [25].

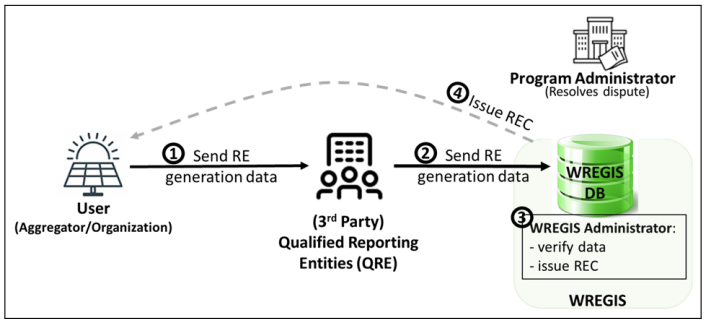


Fig. 5: WREGIS data reporting and REC issuance process.

the producer (micro-generator owner), consumer, local service provider (e.g., Enmax), and the tracking system (e.g., WREGIS). The RE producer (e.g., rooftop solar panel owner) registers with a local utility service provider (SP), supplies excess energy to the grid, and reports the RE quantity to SP. The SP verifies the data from the government authority and issues compensation credit at a preset fixed rate to the RE producer for the supplied electricity. The SP then reports the generation data to a tracking system (e.g., WREGIS) and receives RECs in return (as shown in Figure 5). Finally, the SP sells renewable energy to consumers at its own rate and retires one REC for every 1000 kWh of renewable energy sold.

Issues in traditional micro-generation credit system: Based on WREGIS assumption on its entities (Figure 6), WREGIS operation, the workflow of the traditional micro-generation credit system and the interactions between the involved entities, we have identified the following issues:

1. **Need for a trusted intermediary**: Every WREGIS customer requires a trusted third party, such as a Qualified Reporting Entity (QRE), to report their renewable energy generation data. However, there is a risk that the QRE could collude with an untrustworthy customer to manipulate the reported data. While still meeting WREGIS's minimum standards, they could falsely report a higher generation amount than what was actually produced. As a result, the WREGIS administrator might issue a REC on incorrect data.

Entities	Trust Assumption
Individual Customer	Untrusted
Aggregator	Untrusted
Organization	Untrusted
Qualified Reporting Entity (QRE)	Untrusted
WREGIS Administrator	Trusted
Program Administrator	Trusted

Fig. 6: WREGIS entities and trust assumptions.

2. **Centralized and less user-focused credit system**: Individual or small-scale renewable energy producers must register with a local energy service provider, which acts as an aggregator. The aggregator registers with WREGIS as a customer and reports the combined energy generation data through the QRE. As a result, the REC is issued to the aggregator rather than the original producer. The aggregator then sells the REC at their own price, while the original producer receives credit from the aggregator at a pre-set price.
3. **Lack of exchange facility**: RECs can only be transferred between WREGIS-registered customer accounts. RECs issued to one account for a specific fuel type (e.g., solar) *cannot be exchanged* with RECs generated to another account from a different fuel type (e.g., water). This is important because the ability to exchange RECs from one fuel type for another allows energy producers to optimize their production costs for a specific type of fuel and enables consumers to meet their energy demands based on the available supply from different sources. For example, a solar energy producer might want to exchange the solar tokens for water tokens during periods of low solar generation, such as during winter or cloudy days, to ensure a continuous supply of energy. Conversely, a water energy producer might want to exchange water tokens for solar tokens during dry seasons when water flow is reduced to meet the energy needs. Additionally, this exchange allows both parties to balance their energy production cost and consumption effectively.
4. **Double counting issue**: As per the agreement, a micro-generator must report 100% of its renewable electricity generation to the tracking system it is registered with and claim the RECs associated with it. However, the same micro-generator unit might register with a different tracking system and claim RECs for the same renewable electricity output. Therefore, the same generator can claim RECs from different registries for the same generated green power.
5. **Incorrect data issue**: If the characteristics of a green energy generating unit significantly change and these changes are not reported to WREGIS in an update, or if inaccurate data is submitted, it may result in faulty creation of certificates for that generating unit.

E Smart Contracts (SCs) in Our System

We have two system smart contracts `MetReg` and `DataVer` for the private chain and an application smart contract `GToken` for token generation on the public chain. The abstract of these contracts are given below ((cf. Algorithms 1 to 3).)

Algorithm 1 Abstract `MetReg` smart contract.

contract `MetReg` **{**
modifier (onlyGA, onlyRA);
function `registerMeter` (bytes32 $meterIDHash$, bytes[] $certificate$, bool $isActive$) external onlyGA
function `verifyMeter`(bytes32 $meterIDHash$) external view onlyRA returns (bool)
function `updateMeterStatus`(bytes32 $meterIDHash$, bool $isActive$) external onlyGA
function `verifyGASignature`(bytes32 $certificate$, bytes32 $GAsignature$) internal view returns (bool) **}**

Algorithm 2 Abstract `DataVer` smart contract.

contract `DataVer` **{**
modifier (onlyRA, onlyOracle);
function `submitRECommitment` (bytes32 $RECommitment$) external
function `submitOracleVerification`(bool $isValid$, bytes32 $generationDataHash$, bytes $SPSignature$) external onlyOracle
function `storeVCCommitment`(bytes32 $vcHash$, bytes32 $generationDataHash$) external onlyRA
function `matchVerificationRequest`(bytes32 $userDataCommitment$, bytes32 $oracleResult$) internal view returns (bool) **}**

Algorithm 3 Abstract `GToken` smart contract.

contract `GToken` **{**
constructor (address $Contract\_Owner$);
modifier (onlyRA);
(ERC-20 Token standard functions:) `totalSupply`, `balanceOf`, `transfer`, `allowance`, `approve`, `transferFrom`;
function `requestGToken` (bytes $proof$, bytes[] $disclosedData$, bytes[] $publicParams$) external;
function `verifyBBSProof` (bytes $proof$, bytes[] $disclosedData$, bytes[] $publicParams$) internal pure returns (bool);
function `verifyRequestCommitment`(bytes32 $disclosedDataHash$) internal pure returns (bool);
function `issueToken`(address $ownerID$, uint256 $amount$, bytes32 $AuxiliaryTokenData$) onlyRA public **}**

F Interactions in Green Token Generation

Figure 7 shows the user interaction flowchart in the green token generation process.

G Entity Views and Privacy in Token Generation Protocol

The views of different entities in the token generation process during private and public blockchain interactions are given in Tables 3 and 4, respectively.

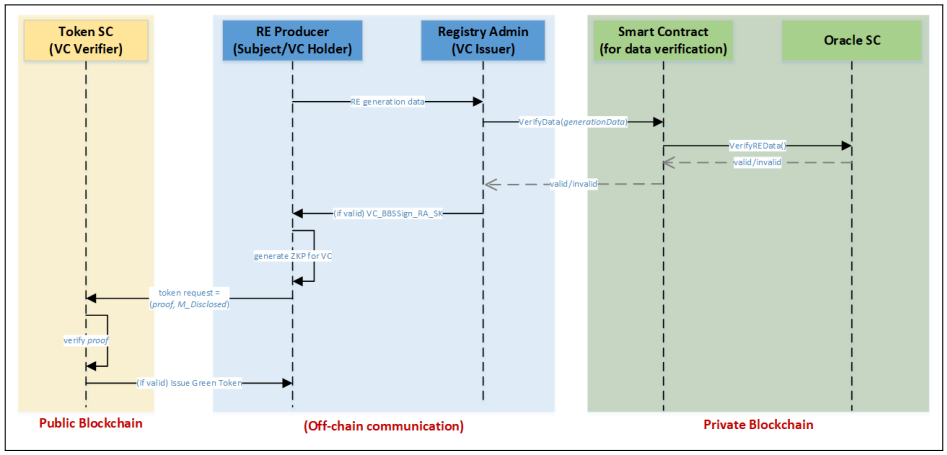


Fig. 7: Green token generation flowchart.

Table 3: View of entities during user interaction in private blockchain.

Entity	Accessible (Visible) Data
Renewable energy (RE) producer	$generationData_{REProducer}$, $VC_{BBSSign_{RA}}$ and public parameters to verify BBS+ signature, Meter certificate(s)
Registry administrator (RA)	$generationData_{REProducer}$, $VC_{BBSSign_{RA}}$ and public parameters to verify BBS+ signature, Meter certificate(s)
Governance authority (GA)	$Enc(generationData_{REProducer})$, Meter certificate(s)
Other Registered users	$Enc(generationData_{REProducer})$
Oracle system	$Enc(generationData_{REProducer})$

Privacy Analysis Detailed Discussion. The privacy analysis of the proposed green token generation system is based on capturing the views of users and other entities in the system and inferring what can be inferred from these views.
View of interaction during token generation protocol: The information visible to each entity during the two interaction scenarios, private and public blockchain interactions (as mentioned in Section 4.1)—are presented in Tables 3 and 4, respectively. Based on these views, we provide our argument regarding the privacy requirements outlined in Section 4.1.
Our token generation protocol ensures that the privacy requirements are fulfilled through a combination of verifiable credentials (VCs), BBS+ signatures for selective disclosure, and a hybrid blockchain model.

1. **Privacy goal (1)** is guaranteed because of the followings:
   a) For the data verification, the registry administrator (RA) submits encrypted RE *generationData* to the Oracle system on the private blockchain, which is decrypted off-chain by the trusted external data source, and only a binary verification result (valid/invalid) is posted on-chain. Once verified, the RA issues a verifiable credential (VC), signed using the BBS+ signature scheme, attesting to the validity of the *generationData*. This VC is sent off-chain to the producer.

b) For meter verification, the RA checks the meter's certification details and validates the governance authority's (GA) signature by querying the `MetReg` smart contract using the hashed $meterID$. Since the $meterID$ is not disclosed, it remains invisible to other registered users.

Thus, during both the data verification and the meter verification processes, the identity of the RE producer and its *generationData* remain hidden from the private chain adversaries (as defined in the threat model in Section 4). Registered users and Oracle system entities can only see the encrypted RE data. As a result, when the producer submits a *token_request* on the public blockchain revealing only a subset of *generationData*, observers (who are also registered users of the private chain) cannot link private chain verification requests to public chain token requests, satisfying the *cross-chain unlinkability* property.

2. **Privacy goal (2)** is guaranteed because the *token_request* includes a *BBS+ proof* derived from a BBS+ signed Verifiable Credential (VC) along with a subset of data from *generationData*. The Verifier (i.e., the `Token` smart contract) checks the validity of the BBS+ proof, proving the user's possession of a valid BBS+ signature from RA (the Issuer of VC) and the disclosed subset of RE data is signed as part of that signature. Validating the proof guarantees the authenticity and integrity of the disclosed and hidden data and the knowledge of the BBS+ signature. Note that, the disclosed data must be supplied to the Verifier in the same order as they were as part of the *generationData* given to the RA for verification. Otherwise, the proof verification will fail. The *selective disclosure* property of the BBS+ signature scheme [37,12] enables the requester to hide sensitive RE data such as $meterID$, $siteID$, or $ownerID$, and reveal only the necessary subset of data (e.g., $timestamp, REType, REQuantity$) on the public blockchain for token generation, ensuring that no observer can link the token request to a specific meter/facility location/RE producer.
3. The use of BBS+ signature scheme ensures **privacy goal (3)** is satisfied. Each BBS+ proof (e.g., a zero-knowledge proof of knowledge of the signature) submitted by a user to the Verifier with a *token_request* is unique. This uniqueness arises from the varying attribute values in *generationData* (e.g., different $REQuantity$ values at different *timestamp*). Additionally, a public blockchain user can employ multiple pseudonyms derived from a single public key to hide their identity. As a result, even if multiple requests come from the same RE producer, public chain observers cannot correlate them, guaranteeing *single-chain unlinkability*.

Table 4: View of entities during a token request in the public blockchain.

Entity	Accessible (Visible) Data
Token requester (VC holder)	$generationData_{REProducer}$ (= {Disclosed data + Hidden data}),*BBS+ Proof* Public parameters for signature and proof verification, *Green Token*, *Requester pseudonym*
Registry administrator (RA)	*token_request* = (*BBS+ Proof*, Disclosed data, Public parameters for signature and proof verification), *Green Token*, *Requester pseudonym*
Governance authority (GA)	*token_request* = (*BBS+ Proof*, Disclosed data, Public parameters for signature and proof verification), *Green Token*, *Requester pseudonym*
Public blockchain observers	*token_request* = (*BBS+ Proof*, Disclosed data, Public parameters for signature and proof verification), *Green Token*, *Requester pseudonym*

H Implementation Details

System Setup. The setup phase involves initializing the necessary components and deploying the necessary smart contracts for the token generation process.
(i) *Blockchain networks*: Our implementation is designed to be general and applicable to any EVM-based private-public blockchain setting. For testing, we used *Ganache* [15], a widely used framework to simulate the EVM and Ethereum blockchain on a local computer.
(ii) *Verifiable credential (VC)*:We implemented verifiable credentials (VC) following the W3C data model [25] in Python (`VC.py`, pseudocode is given in Algorithm 4). The VC includes *credential metadata*, *attributes of renewable energy (RE)* (i.e., claims), and *a digital signature* on the metadata and attributes (as proof), encoded in JSON format. For signature (i.e., proof) generation, we utilized a browser-based BBS+ signature demo [29], which implements core functions such as key generation, signature generation and verification, and BBS+ proof generation and verification in JavaScript. This web-based tool was used to create BBS+ signatures on manually provided metadata and RE attribute values. The resulting BBS+ signature value, along with the credential metadata and RE data, is fed into the `VC.py` program, which outputs a digital credential in JSON format.
(iii) *Smart contracts and deployment*: We implemented the `MetReg`, `DataVer` and `GToken` smart contracts in *Solidity* language using the Remix IDE [19].
(iv) The `GToken` contract, upon receiving necessary RE verification data such as *BBS+ proof*, *disclosed data*, and *public parameters* - (1) validates the zero-knowledge proof (ZKP) generated from the BBS+ signature in VC and (2) mints green tokens upon successful verification. The contract follows the ERC-20 token standard by importing key ERC-20 functions from the OpenZeppelin library [30].
(v) The `MetReg` contract stores meter certification data from the governance authority (GA) and allows the registry administrator (RA) to retrieve data from it for meter verification.
(vi) *Device specification*: We evaluated the performance of our system on a machine running Windows 11 with Intel(R) Core(TM) i7 CPU @ 3.60 GHz and 8 GB RAM.

`GToken` contract implementation challenge and our choice of solution. The `GToken` contract integrates both a BBS+ proof verifier function and an ERC20 token generation functions. The BBS+ proof is a non-interactive zero-knowledge proof (NIZK) and relies on pairing-friendly elliptic curves like BN254 or BLS12-381, which Solidity does not natively support. Additionally, for the proof-of-concept, we use a web-based tool [29] for BBS+ signature and proof generation, which makes it challenging to match and implement the exact proof verification logic within a Solidity smart contract. Instead of implementing the exact BBS+ proof verifier on-chain, we implement a ZKP verification in Solidity using OpenZeppelin's precompiled bn256Pairing library for elliptic curve operations to provide a gas cost estimation for ZKP verification in the public chain.

Algorithm 4 W3C Verifiable Credential (VC) Construction (`VC.py`)

Input: Credential metadata, RE attributes, and BBS+ signature
Output VC in JSON format with computation time measured

$issuerName, issuerID, VCID \leftarrow$ User input
$ownerID, meterID, siteID, REType, REQuantity, timestamp \leftarrow$ User input
$BBSplusSignature, publicKey \leftarrow$ User input
Start Timer T_{start}
  Construct VC:
    Initialize VC as an empty dictionary
    $VC[\text{“@}context\text{”}] \leftarrow$ "https://www.w3.org/2018/credentials/v1"
    $VC[\text{“}id\text{”}] \leftarrow VCID$
    $VC[\text{“}type\text{”}] \leftarrow [\text{”}VerifiableCredential\text{”}, \text{”}RenewableEnergyVC\text{”}]$
    $VC[\text{“}issuer\text{”}] \leftarrow \{\text{”}name\text{”} : issuerName, \text{”}id\text{”} : issuerID\}$
    $VC[\text{“}issuanceDate\text{”}] \leftarrow$ current timestamp
    $VC[\text{“}credentialSubject\text{”}] \leftarrow \{\text{“}ownerID\text{”} : ownerID, \text{“}meterID\text{”} : meterID,$ $\text{“}siteID\text{”} : siteID, \text{“}REType\text{”} : REType, \text{“}REQuantity\text{”} : REQuantity,$ $\text{“}timestamp\text{”} : timestamp\}$
    $VC[\text{“}proof\text{”}] \leftarrow \{\text{“}type\text{”} : \text{“}BBS + Signature\text{”}, \text{”}signatureValue\text{”} : BBSplusSignature,$ $\text{“}publicKey\text{”} : publicKey\}$
End Timer T_{end}
Compute Execution Time: $T_{exec} \leftarrow T_{end} - T_{start}$
Store VC in File:
Open file "VC_record.txt" in append mode
Write VC as JSON format to file
Close file
Display VC and Execution Time T_{exec}