Towards A Blockchain-Based Revenue Distribution Mechanism for Electrical Generators

Bryan Bird
School of Electrical and Electronic Engineering
University College Dublin
Dublin, Ireland
birdbjm@gmail.com

Almero de Villiers
School of Electrical and Electronic Engineering
University College Dublin
Dublin, Ireland
almero.devilliers@ucdconnect.ie

Paul Cuffe
School of Electrical and Electronic Engineering
University College Dublin
Dublin, Ireland
paul.cuffe@ucd.ie

***Abstract*—Renewable energy generators have increased in number in recent years to meet the demand for the green transition. However, the inconsistent nature of the output from these sources, coupled with fluctuation in demand and market spot prices produces significant financial risk for renewable generators. An inflexible market structure exacerbates these effects, and renewable generators may benefit from a remodelled revenue distribution system. A potential solution to these problems could lie in distributed ledger technology, specifically blockchain. Advances in this technology have given rise to smart contracts—autonomous, immutable and transparent computer programs that reside on the blockchain network—that can enable new business models. This paper looks to utilise advances in blockchain smart contract technology to create a novel system for managing revenue streams. The paradigm suggests tokenising the right to a generator's regular market remuneration, creating a new financial tool for stakeholders. A blockchain program consisting of a smart contract is developed, serving as an investigation into the granular workings of such a system. This smart contract manages generators and stakeholders, while directing the movement of tokens and currency. The code is deployed to the blockchain, and tested with simulated transactions to demonstrate the effectiveness of the mechanism. Further development of such a solution, specifically revenue sharing between grouped generators, could offer significant stability to financial models for renewable energy generators.**

Keywords*—blockchain, smart contracts, electricity markets, tokenisation

I. Introduction

RENEWABLE energy generation presents a vital part of the transition towards a fully decarbonised energy supply. These technologies, however, present a myriad of challenges related to variance of supply, due to dependence on weather patterns. This translates into volumetric risk on the generators' parts [1], and they may experience cashflow problems in unfavourable weather patterns. These problems are further exacerbated by current remuneration and financial handling methods available to generators [2]. Often these are inflexible and unsuited to the evolving field of renewable energy.

To these ends, a commonly-used financial instrument is that of the Power Purchase Agreement (PPA). A PPA aims to offset the upfront cost of renewable generation by first receiving a down-payment from the buyer. Buyers are large-scale offtakers, typically commercial entities with significant demand, that enter into contractual agreements to purchase energy at a fixed price for a fixed time [3]. Such PPAs have the disadvantage of being inflexible, serving only to mediate an arrangement between a single buying and selling pair. This shortcoming, coupled with the necessary scale of the offtaker, presents a potential need for more flexible and accessible solutions based around financial handling and distribution for renewable generators.

A growing base of research suggests Distributed Ledger Technology (DLT) as a potential tool. The term refers to a category of decentralised computing, closely related to blockchain technologies. This technology was originally developed as a transparent and secure means of facilitating the transfer and storage of value [4], culminating in the high-profile rise to popularity of cryptocurrencies. A further evolution of blockchain came in the form of *smart contracts*, which built on the technology for additional functionality [5]. The term refers to decentralised computer programs residing on the blockchain which execute under certain conditions without the need for an intermediary [6]. That is, they function as transparent and autonomous financial handling tools, and can oversee the dispersal of native cryptocurrencies to stakeholders without human input. Applications built with smart contracts are commonly known as dApps (decentralised Applications).

Such smart contracts serve as the basis of *Decentralised Finance* (DeFi). This philosophy sees blockchain applications imitating traditional finance mechanisms, while drawing on the accessible and transparent nature of the underlying blockchain basis. DeFi methods have presented evidence as a solution to financial risk [7], specifically, the application of cryptography in the form of blockchain [8]. These applications have been modelled and implemented as a way of optimising financial arrangements for renewable energy generators, particularly in transactive energy applications [9].

In response to the challenges facing renewable generators, and inspired by ideas from the sphere of blockchain and DeFi, this paper presents the concept of Revenue-bearing Tokens (RevToks). First proposed in [10], these are cryptographic assets that represent the inherent right to draw funds from a specific generator's regular market proceeds. Authors in [10] propose a speculative new paradigm for the electricity industry using such tokens to mediate financial relationships between electricity buyers and sellers. The purpose of the current paper is to show the feasibility of this novel system by developing the underlying code and committing it to the blockchain.

In this model, electrical generators are free to *tokenise* a portion of their revenues. In other words, a blockchain dApp allows them to mint RevToks that represent a partial claim to

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 20/EPSRC/3700 and by the Sustainable Energy Authority of Ireland under the SEAI Research, Development & Demonstration Funding Programme 2018, grant number 18/RDD/373

the money they receive for electricity sales. Generators are free to then sell and transfer these RevToks as required. From the point of view of the RevTok holder, their investment resembles a traditional dividend-paying stock. Tokens may be held by all stakeholders, including private investors, financiers, or other generating firms.

In return for a generator's energy production, proceeds are paid out as cryptocurrency by the existing electricity pool market operator at predefined regular intervals. These proceeds, however, are managed and distributed by bespoke smart contracts. The smart contracts observe token ownership, and autonomously distribute funds based on the presence of these tokens in users' blockchain wallets.

In simple terms, an individual receives a pro rata portion of a renewable generator's regular financial outturns by holding that generator's associated RevTok. This arrangement is illustrated in Fig. 1, with red tokens representing RevToks and blue Ether tokens representing the cryptocurrency whereby payments are made.

Work in [10] presented a high-level simulation of such an arrangement whereby generator revenue streams were subdivided and tokenised. The authors applied modern portfolio theory to such tokens, serving as a financial optimisation method for token holders. Results showed evidence as to the benefits of implementing the RevToks paradigm for renewable generators, serving as a means of decreasing their income variance, thus addressing both the price and volumetric risk that renewable generators endure [11]. This serves as an incentive to continue research into tokenised revenue streams as a tool for electrical generators.

The current paper takes a more fundamental approach to the tokenised revenue streams mechanism. The authors continue the narrative of the proposed RevToks-based system by developing the required novel smart contracts for these applications. This serves as an exploration of how token and revenue handling mechanisms may be implemented in a real-world setting. These functions are developed here to address participant and generator handling, as well as RevTok and token distribution. Finally, a test-case is performed, utilising a real blockchain network. Thus, drawing on evidence from previous work, this paper attempts to establish the viability of real-life operation of tokenised revenue stream systems as financial handling and distribution tools in an electrical power context.

II. Outline of Implementation and Assumptions

The most common existing arrangement in power systems see generators selling exclusively to a central pool market operator. Therefore generator incomes stem from a single source; their sole export is bought by a single entity. Although inspired by decentralised concepts from blockchain, the paradigm proposed in [10] and continued here assumes that this existing electricity pool market operator remains in place. Since the source of revenue is trusted, regular and secure, the authors believe that the RevToks-based paradigm is a good fit for this arrangement, and serves to address blockchain's *Oracle* problem [12]. In the proposed arrangement the entity that provide remunerations to electrical generators and manages the provision of electrical energy is almost unchanged [13]. The market operator in existing arrangements is already making monetary transfers on a regular period to some address or bank account. The significant change here comes in the form of *how* the market operator distributes these funds. The proposed RevToks paradigm assumes that novel smart contracts, as developed here, are in place that can receive funds from the pool market operator in the form of cryptocurrency [14], a minor change in how remunerations are settled. These smart contracts manage RevToks and track their ownership, offering token holders the absolute right to draw these funds. Pool market operators will regularly pay out to these smart contracts at predefined times, an obligation that they publicly and credibly commit to. It is assumed that a portion of tokenised funds are paid out the generator i.e. generators hold some of their own RevToks, but this is excluded from Fig 1 for clarity.

The below assumption, adapted from [10], summarise the above points and serve to guide the current technical solution and implementation.

- There exists a central mandatory pool market operator that buys electricity from generators and settles revenues based on energy markets outturns [13], [14].
- Inalienable, self-enforcing claims on these revenue streams can be subdivided and tokenised into RevToks, which can be transferred, traded, and stored [14]–[16].
- The pool market operator sends funds in the form of crypto stable coins to a dedicated smart contract that autonomously and transparently oversees monetary distribution to RevTok holders.

The above assumptions will serve as the template for the current paper's proposed revenue distribution arrangement. Solutions presented here will attempt to implement this model using modern blockchain development standards and practices, demonstrating its potential for real-world application.

The primary focus of this paper is placed on the implementation of a *tokenising* mechanism for revenue of a renewable generator to test the RevTok concepts real-world application. Additional supporting functionality is included, such as groupings for administration, which could allow novel solutions to financial risk and further optimisations of renewable generators' business models. The code developed here also includes the revenue distribution mechanisms that distribute generation proceeds based on RevTok ownership.

III. Methodology

To develop a tokenising mechanism to test the RevToks concept in a real-world scenario, a smart contract solution was tested using modern blockchain development standards and materials.

The solution for a revenue tokenising mechanism for generators is deployed using smart contracts written in solidity with associated testing carried out using JavaScript. This contract utilised the Ethereum Virtual Machine (EVM), a "*virtual state machine that ... is a quasi-Turing complete machine*" [5]. This virtual machine updates the state of the blockchain and allows code to be executed in transactions on a decentralised ledger.

Solidity is a *"statically-typed curly-braces programming language"* stored on a block chain [17] and is used for the development of smart contracts. This code executes statically through function calls, performing computation or storing data in variables. The smart contract code functions both to store and distribute tokenised funds for generators' revenue streams. The *Hardhat* framework [18] was used to manage and deploy this system to a *Goerli* test network, a development environment for the Ethereum network.

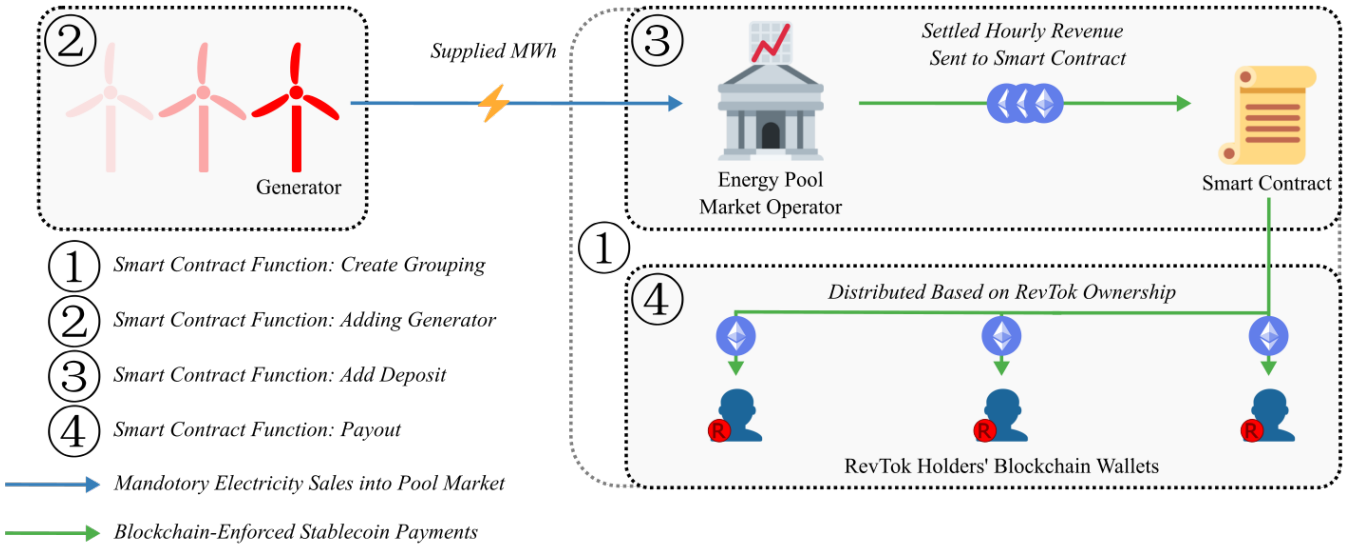

Fig. 1: Regular periodic revenue distribution schemes for electrical generators based on a proposed mechanism in [10]. Ethereum logo used courtesy of [5].

The code makes use of the *OpenZeppelin* [19] library of audited smart contracts to implement some boiler-plate functionality. These open source contracts provide pretested contracts which can be imported and utilised the the current implementation. This is particularly common when working with implemented token standards such as ERC-20 or ERC-1155, which allow for interoperability with other systems. The exact token standard used in this case study is discussed below.

A. Token Standard

This paper utilises the the ERC-1155 token standard, chosen due to its flexibility in deployment and divisibility into unique granular units. The ERC-1155 token is *"a standard interface for contracts that manage multiple token types"* as outlined in [20]. In essence, this means there is a standardised version of this token that everyone agrees to operate by. This token can be either deployed separately or in batches and can be fungible or non-fungible, coupled with an adjustable supply. This, in turn, allows flexibility in deploying different arrangements of investments for different generators. As an example, a unique token may be minted for each individual stakeholders who interact with the system, all stored on the same smart contract.

B. System Overview

This implementation consists of a single smart contract, deploying the ERC-1155 token standard. This single contract will manage all funds and interactions for renewable generators through function calls and stored variables. The system diagram and interactions of this smart contract is outlined in Fig. 1.

All transactions are carried out directly on the Ethereum network, thus, any mention of funds or fees can be assumed to be directly in the native Ether cryptocurrency, unless otherwise stated. The use of Ether is done here for standardised operation on a test-net. Real-world application of such a solution would utilise a stable-coin which is a cryptographic token pegged directly or algorithmically to a fiat currency such as the Dollar or Euro.

For simplicity of testing and demonstration, all transactions will use the lowest denomination of Ether, Wei[1]. This allows for a simpler demonstration of value being moved through the contract with less Ether required while simultaneously ensuring the functionality of all operations.

Function calls are made directly to the deployed smart contract address through the use of Scripts written in JavaScript or through the use of the REMIX IDE [21]. REMIX provides an integrated development environment for deploying smart contracts and a user interface for the blockchain optimising this code.

IV. Results

The deployed implementation seeks to provide a practical implementation of the system outlined in [10] to demonstrate tokenisation of a revenue stream in an electrical power context. This is accomplished by developing a smart contract to facilitate the RevToks arrangement, such that a certain amount of a generator's revenue stream can be tokenised and distributed through an autonomous smart contract. Code for the primary *Revenue Tracking* contract of this system is detailed below, with accompanying demonstration of transactions. This can be viewed deployed on the blockchain and transactions viewed at a block explorer service such as Etherscan.

A. Revenue Sharing Contract

Acting as a central contract for this system, the revenue-sharing contract will handle the management of funds deposited by a market operator for onward redistribution. To do so the contract will make use of the ERC-1155 token standard to allocate tokens in exchange for Ethereum to track revenue for each generator. All funds will be stored at this contract address once deployed to an EVM-compatible chain.

The contract is initialised with a set number of unique ERC-1155 tokens that can be issued when a generator is added to the contract. Initially, the contract operator creates groupings for more granular control of generators and payments. This operator can be a regulated entity, third party entity or public body depending on the market circumstances. Groups are created

[1] 1 Ether = 10^{18} Wei.

using the *createGrouping* function as shown in Listing 2. These groupings will hold basic data, such as an ID, list of developers and current balance of the of the group. Groupings are an additional management feature for the contract's operators to organise generators and respective funds. These groupings are not essential for basic operation of tokenising of a revenue stream rather are supplementary for administrative tasks. This process corresponds to step (1) in Fig. 1.

Listing 1: CREATE GROUPING

```
//Sets up a group and populates a new struct with
    data
    function createGrouping (uint256 _id) public
        onlyOwner{
        //Sets ID for group and balances to 0
        RevGroup storage group = groupings[_id];
        group.id = _id;
        group.balance = 0;
    }
```

Next, the operator will add a generator to this contract by calling the *addGenerator* function. This assigns one of the unique tokens to that generator for tracking of any deposits. A set price is also given for the token, for example a token may be set to equal one Dollar or Euro in the equivalent amount of Ether depending on the market. Finally, the generator is added to a grouping and their balance is initialised to zero. This process is illustrated in step (2) of Fig. 1.

Listing 2: ADDING GENERATOR

```
//Adds a generator to a set group and sets mappings
    for there grouping, tokenID and the price of
    that token.Takes the generator, a group, a new
    token ID and the token price as arguments.
    function addGenerator (address _generator,
        uint256 _group, uint8 _tokenID, uint256
        _price) public onlyOwner{
        RevGroup storage group = groupings[_group];
        //Adds the generator's details to the group
            for storage.
        group.developers.push(payable(_generator));
        tokenID[_generator] = _tokenID;
        price[_tokenID] = _price;
    }
```

A portion of the revenue owed to the generator from electrical generator will be tokenised by calling the *addDeposit* function shown in Listing 3, and demonstrated in step (3) of Fig. 1. This function takes the address of the generator, their assigned group and unique token ID. A transaction calling this function on the blockchain will also contain the associated funds owed for that payment, which will be sent with the function call. The contract then receives the Ether to be held at its address. The function will then update the balances for the generators address within the contract. Finally, the number of the generators unique tokens to be minted and transferred is calculated based on the token price and amount of Ether received.

Listing 3: ADD DEPOSIT

```
//Function for tokenising a fixed amount of revenue
    for a generator.
    function addDeposit (address _address,uint256
        _group , uint8 _tokenID) public payable{
        //investment balances for the generator and
            grouping are updated.
        investment[_address]+=msg.value;
        RevGroup storage group = groupings[_group];
        group.balance += msg.value;
        //amount of tokens to be minted is determine
            from the token price and the value sent
        uint256 amount = msg.value/price[_tokenID];
        _mint(_address,_tokenID,amount,"");
    }
```

A call to the contract will then made using the *payout* function shown in Listing 4. The function receives the generator's address and grouping as arguments. This will transfer the correct amount of funds to the generator's address based on the number of tokens they hold. These values are calculated by retrieving the balance of the ERC-1155 token from the generator's wallet address and calculating the total percentage value of the token supply using native functions of the ERC-1155 standard. This final process corresponds to the labelled (4) in Fig. 1.

Listing 4: PAYOUT

```
//Calls the contract to payout the amount owed to
    the user taking their address and group as
    payment.
function payOut ( address payable _address , uint256
     _group, uint8 _tokenID ) public  {
        RevGroup storage group =groupings[ _group];
        // validates the contract has required funds
        require ( address (this). balance >= group.
            balance," Invalid Transaction ");
        // calculates the amount dividing by group
            length incase of multiple members
        uint256 _balance = balanceOf(_address,
            _tokenID);
        uint256 _price= price[_tokenID];
        uint256 _amount = _balance*_price;
        // Success condition for transfering funds
        (bool success ,)=address(msg.sender).call{
            value: _amount ("");
        require(success, "Failed to send ether");
        // Updates contract balances to reflect
            change
        investment [_address] -= _amount ;
        group.balance -= _amount ;
    }
```

B. Testing of Revenue Sharing Contract

A case study is performed that sees a simulation of a revenue stream and handling through this contract. This demonstrates the process of tokenising revenue and releasing of funds. A four step process, as outlined in Fig. 1, represents the fundamental mechanism whereby the RevToks paradigm operates. Theses four steps are described below, with numbered labels matching the illustrated steps in Fig. 1 and corresponding to the smart contract functions developed in Listings 1 to 4. Associated real-world blockchain transactions for each step are outlined in Table I, numbered by their corresponding steps. Transactions are available via Etherscan.

(1) Grouping for storing and management of generator's data is created to initialise the contract.

(2) Generator is added to the platform, creating an unique token to represent their revenue stream.

(3) Funds owed to the generator from electrical generation are sent to the smart contract address, updating balances and minting new tokens which are transferred to the RevTok holders blockchain address.

(4) Payout is then called and funds are released to the RevTok holders based on their corresponding token balance

Additional arrangements, as discussed in section IV-C, are not demonstrated here, although such functions are possible within the current implementation.

TABLE I:
DEPLOYMENT COST PER CONTRACT

Phase	Step	Transaction	BLOCK ID	Cost in Ether	Wei Transferred
↣ **Set Up**	①	Group Creation by Scheme Operator [22]	8136072	0.000155	
	②	Generator W-S-S Added by Scheme Operator [23]	8136069	0.000243	
↻ **Operation**	③	Remittance of Revenue Earned by W-S-S in Pool by Market Operator [24]	8140074	0.002765	+9278
	④	Withdrawal of 80% of this Revenue by RevTok holder 1 [25]	8136062	0.000120	-7422
	④	Withdrawal of 20% of this Revenue by RevTok holder 2 [26]	8136062	0.000120	-1856

TABLE II:
STOKEHOLDER VALUATIONS

Stakeholder	Portion of RevToks Held	Wei Produced/Claim
Generator W-S-S	-	9278 Produced
RevTok holder 1	80%	7422 Claimed
RevTok holder 2	20%	1856 Claimed

All transactions shown above according in real-time on the Ethereum test-net using function calls and blockchain addresses. Beyond the use of real Ether or stable coin assets any practical implementation would follow the same procedure. ① and ② constitute the setup of these contracts and the addition of generators. ③ and ④, showcase the main functionality of the system, revenue being deposited, tokenised and transferred to RevTok holders and that revenue being transferred to these holders. In practice these latter steps would be repeated regularly over the systems lifespan, tokenising revenue and transferring funds.

For the purposes of demonstration a revenue transfer and withdrawal was tested with these transactions using data from [10]. Generator W-S-S, a 20MW windfarm based in the West of Germany was used as case study. Its total energy production for the 30th of November 2015 was 348Wh, and this generated a total of revenue of €9278 to be paid out by the mandatory pool market operator. As previously mentioned, for testing purposes 1 Wei will be assumed to be €1 of value in all transactions sent to the contract, due to current economic conditions of the test-net [27].

At the end of this same day withdrawal transactions were called by the two RevTok holders who hold fractional claims on this enduring revenue stream emanating from the mandatory pool market. *RevTok holder 1* held 80% of the supply of RevToks while *Token holder 2* held 20%. Table II shows the amount of revenue paid out in respect of this generator by the pool market operator, along with the claim each RevTok holder has on the total tokenised revenue stream. Notably, the RevTok holders are anonymous, and it practise a generator such as W-S-S may wish to themselves hold a fractional share of their own revenue stream.

Table I describes the practical function, including the *Set Up* phase where financial mechanisms are put in place, and the *Operation* phase that sees the revenue tokenisation and distribution carried out as in the case study. The transaction costs of the exchanges are included, as well as the volume of funds transferred in each steps. Step numbers remain consistent with Fig. 1.

The current system demonstrates the potential simplicity of this financial mechanism, from a generator's perspective, when implemented in Solidity on a compatible blockchain. Financial benefits can be seen from such an arrangement. Total costs associated with the above operations, including the initial deployment of the contract amounted to 0.006768 Ether or approximately $8.88 at the time of writing.

C. Extended Functionality

The system outlined above allows for certain extended functionality and flexibility inherent in its design. These features are non-essential for the mechanism of tokenising a revenue stream but outline novel use cases for managing financial arrangements beyond revenue management.

In particular, the creation of groupings for administrative purposes, as mentioned previously, is one such feature. Groups will be created to manage investments of generators. These groupings will hold basic data, such as an ID, list of developers and current balance of the of the group. Sets of generators can then be easily grouped or revenue split in novel ways among sets of generators allowing for investment pooling or more complex financial arrangements between generators. Full functionality for investment groups of generators has been implemented in this contract in line with the models discussed in [10]. These transactions were not shown here as it is beyond the scope of the current discussion. Finally, due to the fungible nature of these assets and use of a public blockchain, further reselling arrangements can be utilised. Examples of such arrangement could included novel arrangements of generation contracts or investment opportunities [28]. The current implementation of a tokenising mechanism in the form of a smart contract offers support for the development of this extended functionality.

V. CONCLUSIONS

Renewable energy generators are an important part of combating climate change. However, the intermittent nature of their generation makes them subject to cash-flow risk. This is exacerbated by an inflexible market structure, with current hedging and remuneration methods being impractical and unsuited to the unprecedented nature of these non-dispatchable generators. Therefore, opportunity exists for a new revenue handling mechanisms that could serve to accommodate the unique needs of these generators and indirectly incentivise renewable energy investment.

Inspired by developments in the world of DeFi and cryptocurrencies, this paper develops a revenue distribution method based on blockchain smart contracts. The paradigm proposed

here suggests tokenising the regular monetary remunerations of large-scale electrical generators into *RevToks*, or Revenue-bearing Tokens. These RevToks serve as inalienable rights for token holders, in the form of stakeholders or investors, to freely draw pro rata shares of the regular periodic incomes from electricity sales. In this model the central electrical pool market operator remains in place, serving to the source of payment. However, this market operator credibly commits to the transparent use of a smart contract as means of distribution of funds.

Based on previous work, this paper developed the smart contract required in the implementation of the RevToks system described above. Thereby this paper served as an exploration into the more fundamental workings of the system. The smart contract deployed in the above example has the functionality necessary to operate a tokenised revenue stream with real-time modelling necessary to allow for the integration of current business models. The novel smart contract was designed so as to manage stakeholders, including generators and token holders. It also oversees the tokenisation of regular payments from the pool market operator (as electricity sales), and the autonomous distribution of these funds to the respective token holders. The smart contract was tested on a real blockchain network, and showed evidence of facilitating seamless autonomous payments, observing the presence of tokens and enabling owners to claim funds.

The above demonstration highlighted the effectiveness of tokenising revenue streams for renewable generators. The current implementation of a revenue-sharing mechanism based on a tokenised income thus has benefits over traditional financial arrangements, particularly within power systems. The case study, however, did not show the extended ability of the paradigm for managing groups of investors and sharing revenue through the smart contract. Further development of this deployment can utilise these features by creating a front-end management system to integrate and feed calculated data to the smart contract for processing. The RevToks method can thus serve as a hedging mechanism for the reduction of volumetric and price risk on the part of generators, or may allow offtakers to offset their own electricity costs.

Future development of this solution and implementation should attempt to integrate it more closely with existing systems. This could be achieved by developing a usable user interface, along with the addition of other serving oracle contracts that can provide working data on prices and generation. These developments would allow for easier adoption of this solution within the business model of many renewable generators.

References

[1] Z. Guo, W. Wei, M. Shahidehpour, Z. Wang, and S. Mei, "Optimisation methods for dispatch and control of energy storage with renewable integration," *IET Smart Grid*, vol. 5, no. 3, pp. 137–160, 2022. DOI: https://doi.org/10.1049/stg2.12063. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/stg2.12063. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/stg2.12063.

[2] M. Hain, H. Schermeyer, M. Uhrig-Homburg, and W. Fichtner, "Managing renewable energy production risk," *Journal of Banking & Finance*, vol. 97, pp. 1–19, 2018, ISSN: 0378-4266. DOI: https://doi.org/10.1016/j.jbankfin.2018.09.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378426618301882.

[3] R. Proctor, B. Fox, and D. Flynn, "Reserve trading within power purchase agreements," in *DRPT2000. International Conference on Electric Utility Deregulation and Restructuring and Power Technologies. Proceedings (Cat. No.00EX382)*, 2000, pp. 137–141. DOI: 10.1109/DRPT.2000.855652.

[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[5] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, 2014.

[6] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, and F.-Y. Wang, "Blockchain-enabled smart contracts: Architecture, applications, and future trends," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2266–2277, 2019.

[7] H. Natarajan, S. Krause, and H. Gradstein, "Distributed ledger technology and blockchain," 2017.

[8] N. El Ioini and C. Pahl, "A review of distributed ledger technologies," 2018.

[9] D. L. Dinesha and P. Balachandra, "Conceptualization of blockchain enabled interconnected smart microgrids," *Renewable and Sustainable Energy Reviews*, vol. 168, p. 112 848, 2022.

[10] A. de Villiers, J. Byrne, and P. Cuffe, "A diversified portfolio of tokenised revenue streams can provide hedging opportunities for renewable electricity generators," In review, 2023.

[11] S. Deng and S. Oren, "Electricity derivatives and risk management," *Energy*, vol. 31, no. 6, pp. 940–953, 2006, Electricity Market Reform and Deregulation, ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2005.02.015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544205000496.

[12] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," *IEEE Transactions on Software Engineering*, vol. 41, no. 5, pp. 507–525, 2015.

[13] L. M. Cardenas Ardila and C. J. Franco Cardona, "Structure and current state of the wholesale electricity markets," *IEEE Latin America Transactions*, vol. 15, no. 4, pp. 669–674, 2017. DOI: 10.1109/TLA.2017.7896393.

[14] K. Malinova and A. Park, "Tokenomics: When tokens beat equity," *Available at SSRN 3286825*, 2018.

[15] E. Travia. "The coded income model." (2020), [Online]. Available: https://medium.com/coinmonks/the-coded-income-model-81095534e624.

[16] B. Lio. "Revenue-sharing-token." (2020), [Online]. Available: https://smithandcrown.com/glossary/revenue-sharing-token/.

[17] E. Foundation. "Solidity." (2022), [Online]. Available: https://docs.soliditylang.org/en/latest/index.html.

[18] N. Foundation. "Hardhat framework." (2022), [Online]. Available: https://hardhat.org/hardhat-runner/docs/getting-started.

[19] O. Foundation. "Openzeppelin contracts." (2022), [Online]. Available: https://openzeppelin.com.

[20] "Eip-1155 multi token standard." (2017), [Online]. Available: https://eips.ethereum.org/EIPS/eip-1155.

[21] REMIX. "Remix integrated development environement." (2022), [Online]. Available: https://remix-project.org/.

[22] "Transaction for the creation of an investment group." (2022), [Online]. Available: https://goerli.etherscan.io/tx/0xf64fe5046bbae8db3692e8f7921d2d7a00304939b6f31feb8c7a9ee7bc253708.

[23] "Transaction for adding a generator to the smart contract." (2022), [Online]. Available: https://goerli.etherscan.io/tx/0x8149885b53362621f8208623d802a1e214ea0e14f5dd0d600157259a2893e33f.

[24] "Transaction for depositing revenue to the smart contract." (2022), [Online]. Available: https://goerli.etherscan.io/tx/0x7e29099ff99a65485fde9fecc9c421d11e8b2f613cadb651a001c35a8d5b0ce0.

[25] "Transaction for withdrawing revenue from the smart contract." (2022), [Online]. Available: https://goerli.etherscan.io/tx/0xb63355e60e1edb693e84382b4d7eab2d05e211b795588d2f64376424bd9926de.

[26] "Transaction for withdrawing revenue from the smart contract." (2022), [Online]. Available: https://goerli.etherscan.io/tx/0x73c35227cee3ff89352827d3603a78aa3faf01e0578251f284059dc427efc089.

[27] S. Haig. "Shortage of goerli testnet ether puts the squeeze on devs." (2022), [Online]. Available: https://thedefiant.io/devs-goerli-shortage-ethereum.

[28] O. Alao and P. Cuffe, "Implementing contract-for-difference arrangements for hedging electricity price risks of renewable generators on a blockchain marketplace," *IEEE Transactions on Industrial Informatics*, pp. 1–10, 2022. DOI: 10.1109/TII.2022.3185661.