A Decentralised E-Learning System for Micro-Credentials

Roderick Vella Galea
roderick.vella@mcast.edu.mt
The use of ChatGPT3.5 throughout the writing of this report is an experiment

Abstract

This paper introduces a decentralised system that allows educational institutions to issue micro-credentials signed by students and later verified by potential employers. The system is designed with the core principle of operating in a decentralised environment without the need for a central authority. It runs on an EVM blockchain, providing secure and tamper-proof storage of the awarded credentials. As a result, the system can be accessed by institutions, students, and employers globally, without registration or reliance on third-party servers, leading to reduced costs and an uptime guarantee. The system also relies on open-source code, providing transparency and avoiding hidden costs typically associated with proprietary systems. This paper provides a detailed description of the underlying framework for the system.

1 Introduction

Micro-credentials are concise, specialized credentials that demonstrate a person's proficiency in a particular field by showing their in-demand knowledge, skills, and experience. These digital certifications, often known as badges, are usually given in bite-sized formats, allowing individuals to demonstrate mastery in a particular area. The Council of the European Union [1] emphasizes that micro-credentials cannot achieve their full potential without shared standards, transparency, cross-border comparability, recognition, and portability.

With the rise of e-learning, individuals can acquire numerous micro-credentials from different institutions. However, the proliferation of different regulations and practices for awarding and verifying micro-credentials creates challenges for individuals and employers. In particular, it can be challenging for employers to verify the authenticity of the certifications and for individuals to manage their multiple micro-credentials.

This paper introduces an application called txtAwards, specifically designed to provide a standard for awarding and verifying whether an individual truly owns declared micro-credentials. This research focuses on the portable aspect of micro-credentials and ensuring that accomplishments are borderless and accessible from anywhere. The tool makes it simple for educational institutions to work with, encouraging them to give students several digital awards throughout the course. This strategy will eventually motivate students to study more because they are constantly praised for their accomplishments.

Notably, the system allows multiple institutions to issue awards to the same student using the same standards. The application allows students to manage their micro-credentials more effectively and for employers to evaluate the skill set of potential employees confidently. The following chapters cover the application's foundational framework and the suggested system's potential advantages.

2 Overview of the txtAwards application

According to Andreas Antonopoulos [2], the five pillars of open blockchains are (a) open, (b) borderless, (c) neutral, (d) censorship-resistant, (e) public. txtAwards is a web3 system built on these fundamental tenets. The application is open-source and can be downloaded and installed on any desktop platform, without any specialised software required. Registration with a third party is not necessary, and the system can be accessed by anyone from any location, making it borderless. Additionally, the system is neutral and censorship-resistant, allowing any institution to award its students as they see fit, without any external interference.

All the awards are saved as plain text on the blockchain and can be easily verified and read by anyone. The main goal of the application is to provide an inexpensive way for institutions to issue multiple micro-awards to their students, while maintaining the principles of a truly decentralised application. The application is intended for use by three distinct user types: educational institutions, students, and employers. Educational institutions are responsible for issuing and publishing awards, while students sign and accept these awards. Employers are then able to verify the ownership of these awards to ensure that candidates possess the requisite skills and experience.

3 Design and Architecture

The system is designed to use smart contracts to store awards on a blockchain, enabling all user types to have direct access to the blockchain without the need for a centralised authority. By removing a central authority, the proposed system increases complexity but maintains the principles for a truly decentralised application. The process consists of two main sections: the publication of awards and the verification of ownership. For this to work, educational institutions and students must have an Ethereum address. A private key is randomly generated via an offline built-in wallet and is protected by a password for access. Institutions require ether in their account to cover gas fees.

To start, educational institutions deploy a storage contract on the blockchain, allowing them to store awards for their students. The storage contract's address must be publicly published on the institution's website to identify it as their belonging. An award consists only of a title, issue date and the student's public address. No names or ID cards are stored on the blockchain. Students are identified only by their public key generated by the built-in wallet. It is critical that institutions receive permission from the student before publishing any plain text data on the blockchain, as stored data is persistent and cannot be deleted or modified once saved.

When an institution creates an award for a student, they enter the award's title and issue date, and the system produces a file that must be shared with the student, typically via email. The student must sign the award using their private key, indicating that they accept the contents of the award and allowing the institution to publish it on the blockchain. Once the student returns the signed file, the institution publishes the award on the blockchain. Before storing the data on the blockchain, the smart contract checks the signature's validity. If the signature is valid, the award is stored under the supplied student's address. This process is illustrated in Figure 1.

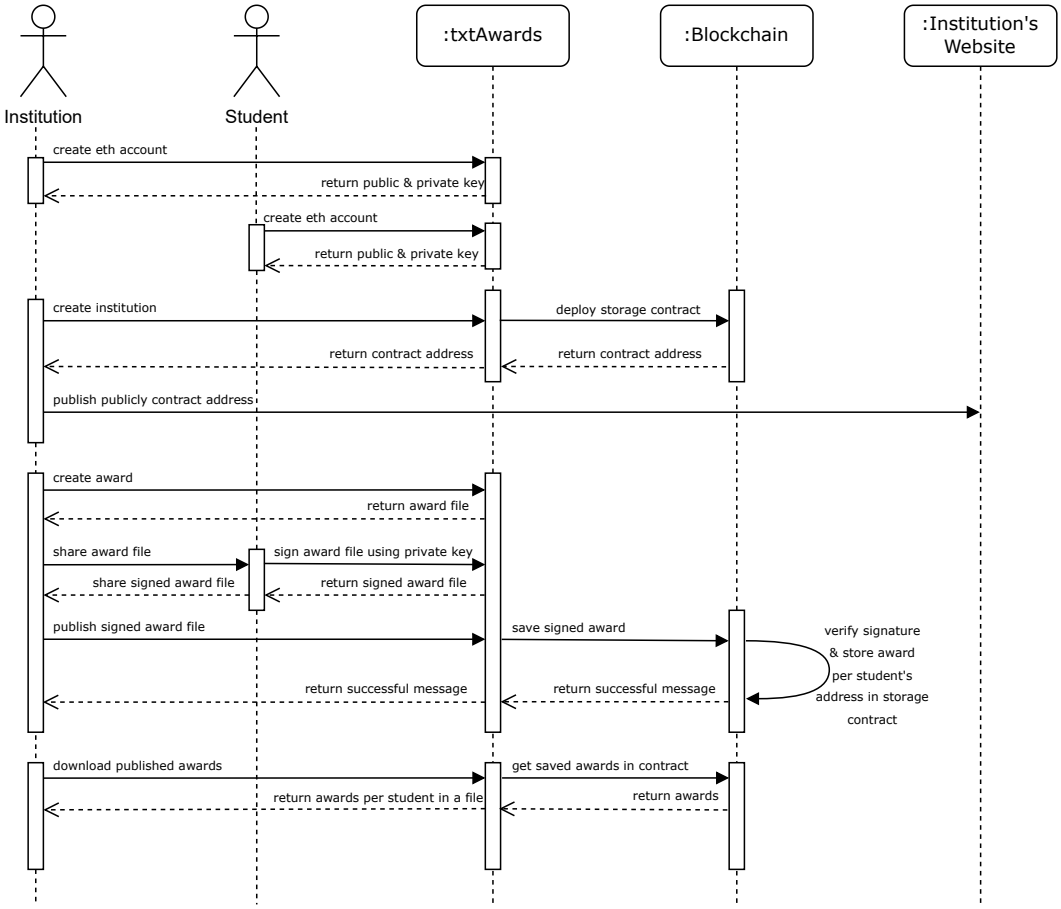

Figure 1: Publication of Awards

The txtAwards system allows a student to receive awards from multiple educational institutions, each with their own separate storage contract. When a student wishes to share their awards with potential employers, the system ensures that the student cannot make false claims by using another student's address to claim ownership of those awards. To assert this, the student is required to sign a custom message for the employer, which can include private information that identifies the student's identity. One has to note that this data is not published anywhere by the system and this process is done offline. The awards, along with the custom message, are then sent to the employer, typically via email.

The employer can use the txtAwards system or public online tools to reverse-engineer the address of the person who signed the custom message. This serves as proof that the person who signed the custom message is the rightful owner of the awards associated with that address, as only the owner of the private key associated with the address can sign a message with that address. This ensures that employers can verify the authenticity of a student's claimed awards without relying on a centralised authority. Finally, the employer can check the institution's identity by checking the published contract's address on their website. This process is illustrated in Figure 2.

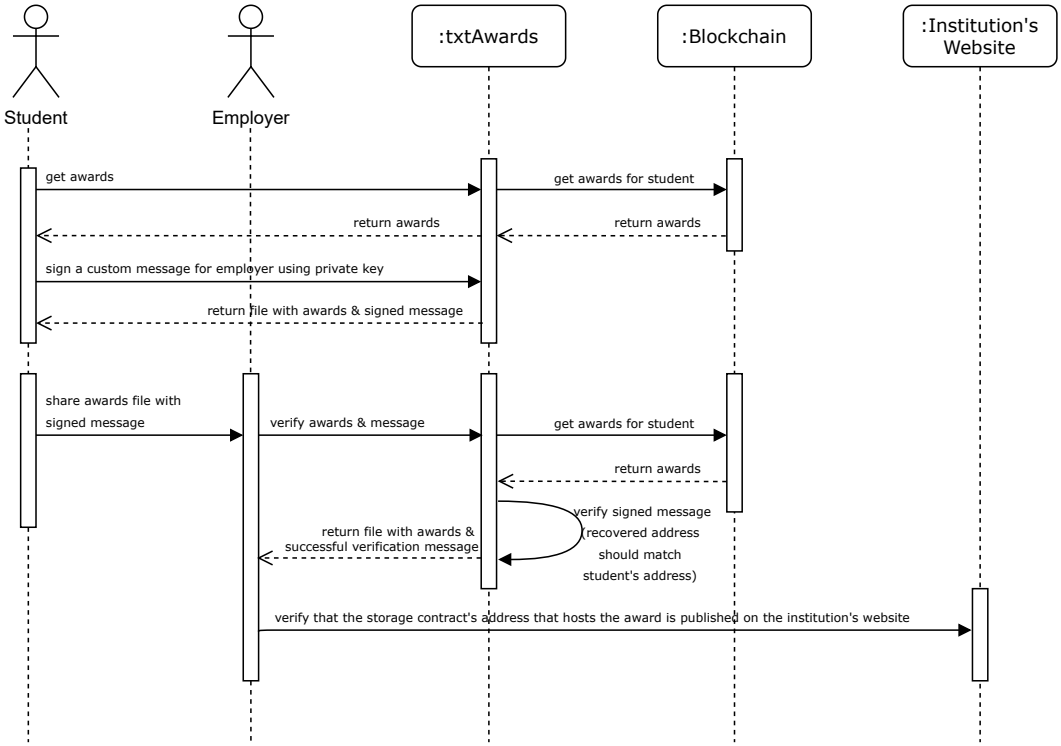

Figure 2: Proof of Ownership

4 Technical Details

The application is written in Python to make sure it runs on all desktop operating systems (Windows, macOS, Linux). CustomTkinter [3] was mainly used for the GUI and Web3.py [4] for interacting with an Ethereum Virtual Machine (EVM) Blockchain. The application also makes use of Solidity for the smart contracts. Hardhat [5] was used to compile and deploy contracts locally.

The system was primarily designed to work with the Polygon blockchain and makes use of Infura [6] to connect. Polygon is a Layer 2 scaling solution for Ethereum that aims to improve its scalability and efficiency. It uses a combination of technologies such as Proof of Stake and Plasma side chains to enable faster and cheaper transactions [7]. For our application, we want educational institutions to award multiple times their students to increase their motivation to continue studying. Therefore, low transaction costs are vital and that's why Polygon was chosen. In the case an educational institution does not want to work with Polygon, then the system can easily be forked to work with other chains as long as they are EVM compatible. Obviously, this will increase the complexity, because a student might have different awards on different chains. The system was designed with this in mind. In fact, the ChainID is always taken under consideration and visible to all user types.

The application has a built-in wallet. This was primary done to reduce the complexity and the dependency on popular wallets such as MetaMask. The wallet is not designed to be used as a general-purpose wallet, but only to process transactions provided by the application. So, it is very limited in design. The Eth-keyfile library [8] was used to handle and store the generated Ethereum private keys and the Polygon Gas Station API [9] was used to get gas price recommendations. This was used to calculate transaction costs before sending them to the Polygon network. The user has to choose from three transaction priority speeds: safeLow, standard and fast.

The system makes use of two smart contracts. The main contract manages the institution's processes such as ensuring that an award is legitimately signed by the student. The other contract acts as an external storage contract for storing awards. The external storage was implemented because storing awards directly in the main contract can cause issues during updates. When an upgrade is required and a new version of the contract is deployed, the old version of the contract still exists on the network and all data, including awards, remain stored at the old contract address. Attempting to migrate this data to the new version can be extremely expensive due to the high cost of writing to storage on the blockchain. Additionally, the process of migrating the awards to a new contract would need to be planned and executed during creation time, which can add complexity to the application. To avoid these issues, the external storage pattern, as described in [10] was followed. This involves separating the storage from the main contract logic by creating a separate contract solely for the purpose of storing awards. This external storage contract should be flexible to avoid the need for structural upgrades, and is designed to last the entire lifetime of the initial contract, hence the name external storage. By using an external storage contract, awards can be easily migrated to a new contract without the need for expensive and complex data migration processes. On another note, the storage contract makes use of EnumerableSet by OpenZeppelin where elements are added, removed, and checked for existence in constant time (O(1)) [11].

In this application, an important security feature is the requirement for a student to sign their own award before it can be published by an institution. This process involves multiple security checks to ensure that the signature can only be used once for a specific contract on a specific blockchain. The code in Figure 3 demonstrates these checks in action. The method checks that the provided signature matches the award title, date, nonceAwardSign, chainID, and the contract's address. The nonceAwardSign is utilized to ensure that the signature can only be used once, which helps prevent re-entry attacks. Additionally, the chainID and address(this) ensure that the signature is only valid for that specific blockchain and contract. Overall, these security measures help ensure the integrity of the system and protect against potential attacks.

```
function addAwardSignedByStudent(bytes memory signature,
                                 address studentAddress,
                                 string memory awardTitle,
                                 uint awardDate)
                                 external onlyOwner{

    bytes32 hash_data = keccak256(abi.encodePacked(awardTitle,
                                  awardDate,
                                  nonceAwardSign[studentAddress],
                                  block.chainid, address(this)));

    bytes32 digest = ECDSA.toEthSignedMessageHash(hash_data);
    address recoveredSigner = ECDSA.recover(digest, signature);

    //recovered address from signature must match student's address
    require(recoveredSigner == studentAddress);

    //increase nonce signature to prevent re-entry attacks
    nonceAwardSign[studentAddress]++;

    //proceed and store award
    storageAwards.addStudentAward(studentAddress,awardTitle,awardDate);

}
```

Figure 3: Add Signed Award - Solidity

5 Limitations

As this system is decentralized and not controlled by a centralised authority, it is more complex to use compared to traditional systems. For the application to be effectively used, all parties involved must have a basic understanding of the system's functioning. Educational institutions and students may need to allocate time and resources to learn how to use the application correctly.

Impersonation attacks are another challenge in this system. A student could create a fake institution and issue multiple awards in their name. It is crucial to conduct proper research on the awarding bodies to verify the ownership of the awards. This process may require manual checking, which can be time-consuming.

Security concerns also exist, particularly if a student's private key is compromised. In such a scenario, an attacker could gain access to their awards and use them for fraudulent purposes. In the worst-case scenario, if an institution's private key is compromised, the attacker could issue awards to students fraudulently. It is, therefore, necessary to take appropriate measures to safeguard the private keys of students and educational institutions.

6 Supplemental Materials

For transparency, we have made the source code for the system available on GitHub. The GitHub repository includes detailed instructions on how to test and run the application. Additionally, we have included a video demonstration that provides a quick overview of how users can interact with the system.

- GitHub Code: https://github.com/roderickvella/txtAwards
- Video Demonstration: https://youtu.be/HtQByfBWKdA

7 Conclusion

In this paper, we have explored a decentralised system for storing educational micro-awards without the need for oracles or centralised authorities. This system is open for adoption and can be used by any educational institution without the need for auditing or permission, which eliminates any hidden costs that institutions might incur with other similar applications.

One of the major benefits of this system is that a single application can be used by the students to receive and manage awards from multiple institutions, making it easier to organize and track their achievements. Additionally, students can receive awards from online courses across the world, as well as classroom-based courses, providing more opportunities for their portfolio.

However, there are limitations to this system, including the need for all parties to have a basic understanding of how the system works. This may require some time investment and resources to learn how to use the application effectively. Therefore, pilot tests need to be conducted to test the system and gather feedback for further improvements.

References

[1] "A european approach to micro-credentials." [Online]. Available: https://education.ec.europa.eu/education-levels/higher-education/micro-credentials

[2] A. Antonopoulos, "The five pillars of open blockchains," May 2019. [Online]. Available: https://www.youtube.com/watch?v=qlAhXo-d-64

[3] T. Schimansky, "Tomschimansky/customtkinter: A modern and customizable python ui-library based on tkinter," Feb 2023. [Online]. Available: https://github.com/TomSchimansky/CustomTkinter

[4] "Web3.py." [Online]. Available: https://web3py.readthedocs.io/

[5] "Ethereum development environment for professionals by nomic foundation." [Online]. Available: https://hardhat.org/docs

[6] Infura, "Infura documentation." [Online]. Available: https://docs.infura.io/infura/

[7] Polygon, "Polygon documentation." [Online]. Available: https://wiki.polygon.technology/

[8] Ethereum, "Ethereum/eth-keyfile: Tools for handling the encrypted keyfile format used to store private keys." [Online]. Available: https://github.com/ethereum/eth-keyfile

[9] Polygon, "Polygon gas station." [Online]. Available: https://wiki.polygon.technology/docs/develop/tools/polygon-gas-station/

[10] "External storage." [Online]. Available: https://fravoll.github.io/solidity-patterns

[11] OpenZeppelin, "Enumerableset." [Online]. Available: https://docs.openzeppelin.com/contracts/3.x/api/utils%23EnumerableSet