Implementation of secure transaction

Nathan Lesourd

Abstract

In this report you will find the design and implementation of a secure transaction management system using blockchain. In order to be able to interact easily with this system, a graphical user interface has been developed. The aim of this project is to understand in depth how the blockchain works, as well as other key cryptographic concepts such as hashing and asymmetric encryption. We will study the strengths and weaknesses of this system to find out in which context the use of blockchain is optimal.

I. Introduction

In the constantly evolving digital age, the security of online transactions has become a major issue. For some years now, blockchain has revolutionised the way we design and secure exchanges of value in an increasingly connected world.

A. Blockchain

Blockchain is a concept that has gained worldwide recognition for its role in the creation and management of the Bitcoin cryptocurrency. However, its potential extends far beyond digital currencies. Blockchain is a decentralised ledger technology that offers a revolutionary way of recording and verifying transactions. Unlike centralised systems, where a trusted entity is required to validate transactions, blockchain enables secure transactions between parties without the need for a trusted intermediary. This technology is based on the fundamental principles of decentralisation, transparency and security to guarantee data integrity and mutual trust between participants [1].

The structure of the blockchain is simple. It is made up of a chain of blocks, each containing several transactions. The blocks and transactions have characteristics such as an identifier, the name of the creator, the creation date and the data. This information makes it possible to obtain a unique entity, which will subsequently be crucial for protecting the content of transactions.

B. Hashing

Hashing is a key process in cryptography that transforms data into a series of alphanumeric characters of a fixed length, known as a hash. The hash operation is designed so that apparently similar data will produce completely different hashes. This property makes hashing ideal for verifying the integrity of data on the blockchain. Transactions are hashed and recorded in blocks, making it possible to detect any subsequent alteration or falsification. Hashing ensures that the data in the blockchain remains immutable and tamper-proof.

The state of the art highlights a technical solution for implementing the hashing system, the SHA256 algorithm. It replaces a method that is now erroneous, namely MD5 [2]. To produce the hash, we concatenate all the elements of the transaction and then apply the hash function. In this way, we create a digital fingerprint of the entire transaction to ensure its integrity. The same process is applied to the block.

C. Cryptography

Cryptography plays a central role in blockchain by ensuring the confidentiality, authenticity and integrity of data. It is used to encrypt information to protect its confidentiality, to create digital signatures to authenticate users and to guarantee data integrity using hash mechanisms. Cryptography is the mechanism that supports all blockchain technology, enabling participants to validate transactions and maintain the security of the network without the need to trust a third party.

In the system we are going to develop, all blocks and transactions have a digital signature. This is generated by performing an RSA encryption on the hash of the block or transaction, using the private key of the person who created it. Thanks to this process, all the nodes in the network can check the integrity of the block or transaction by decrypting the digital signature using RSA and the public key of its creator. If the decrypted digital signature matches the whole content of the block or transaction, this means that there has been no tampering.

II. Design and Implementation

Based on these three major concepts, in the rest of this project we will be implementing a transaction management system using the blockchain to ensure the integrity of transactions.

Initially, a design phase was necessary to define the scope of the work and ensure that we did not forget any of the functionalities included in the specifications. We can distinguish two parts to this project, the back-end with the implementation of the blockchain system and the front-end with the graphical user interface. Initially, we will develop the back-end and then use these methods to develop the front-end.

To design the back-end, I created a class diagram to structure the three major elements of the blockchain: the node, the block and the transaction. These three elements can be identified by several characteristics modelled by attributes in our diagram and carry out several actions modelled by methods. Below is a class diagram of these three elements using the UML formalism.

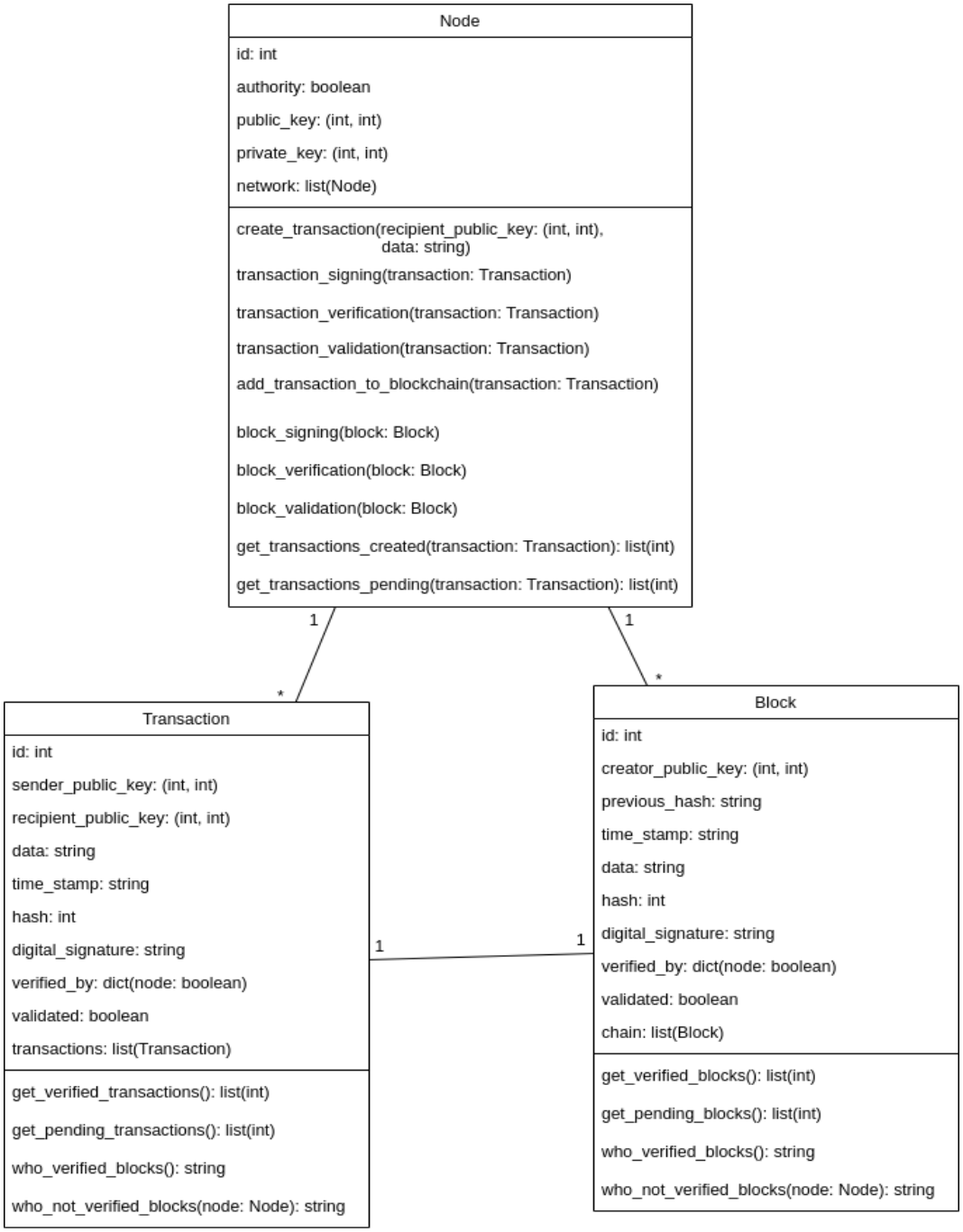

Fig. 1. Class diagram

Using this class diagram, the implementation of the three classes was simplified and then validated using unit and functional tests with Pytest.

Once the business logic had been developed, I got down to creating the graphical user interface. This was something I'd never done before using Python. After some research, I chose to create the front-end with the help of the Custom TKinter API. This library makes it easy to create an interface with a modern layout. You can also find a number of templates for starting your interface. From this GitHub $https://github.com/TomSchimansky/CustomTkinter/tree/master/examples$, I used the image_example.py code as support.

The definition of the template was based on the instructions in this report, which explicitly presents the contents of the various drop-down menus. A flow diagram was drawn up to clarify the architecture of the graphical interface.

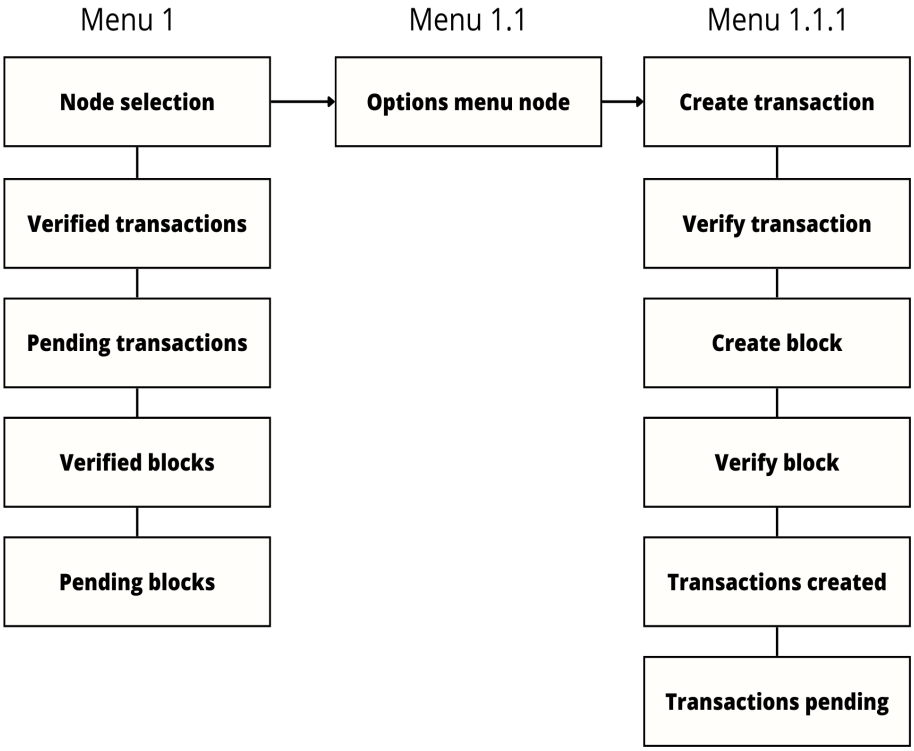

Fig. 2. Flow diagram

III. Test Results

In this section we present the graphical user interface through different scenarios to highlight all the available functionality.

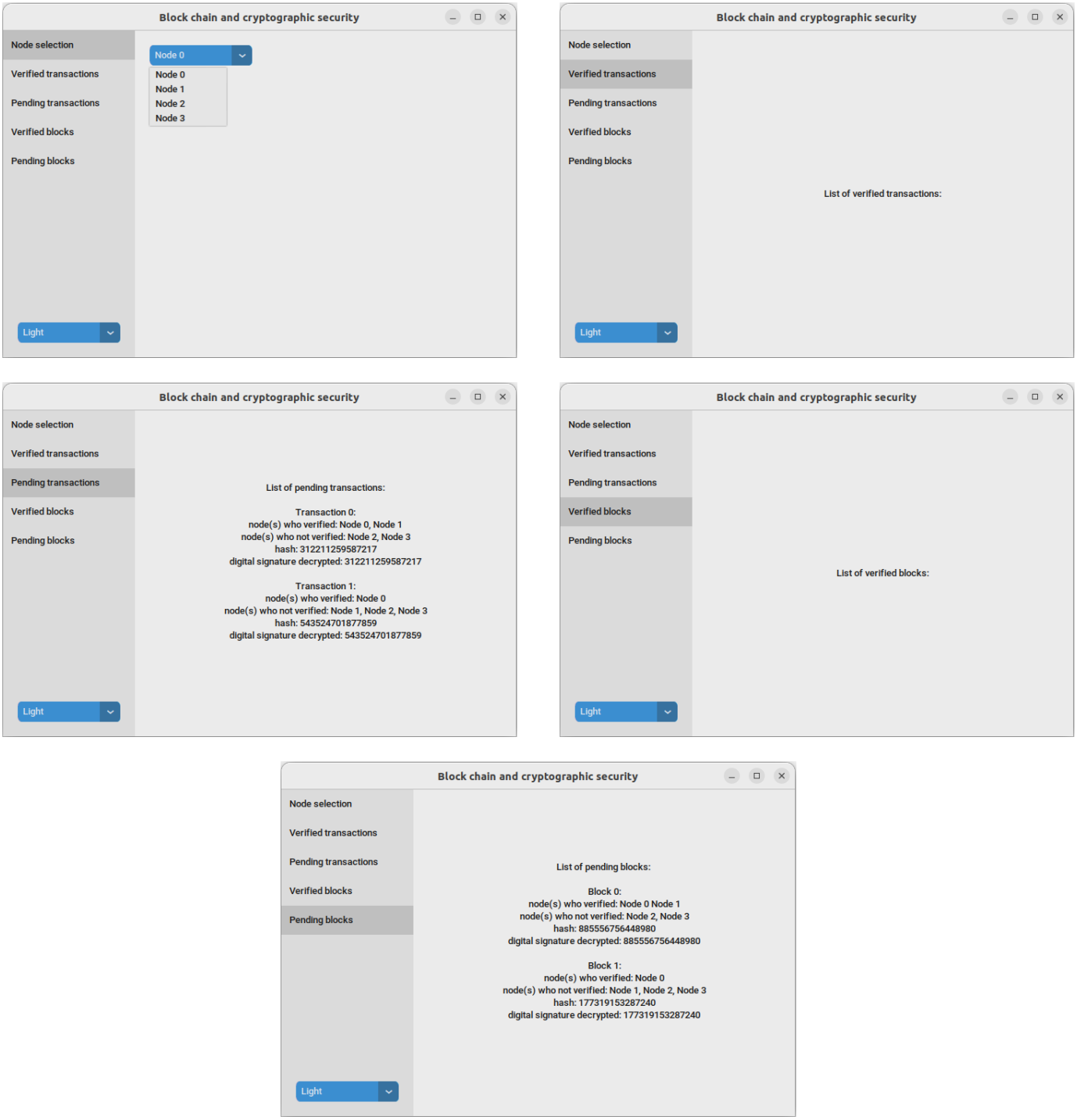

Fig. 3. Graphical User Interface Menu 1

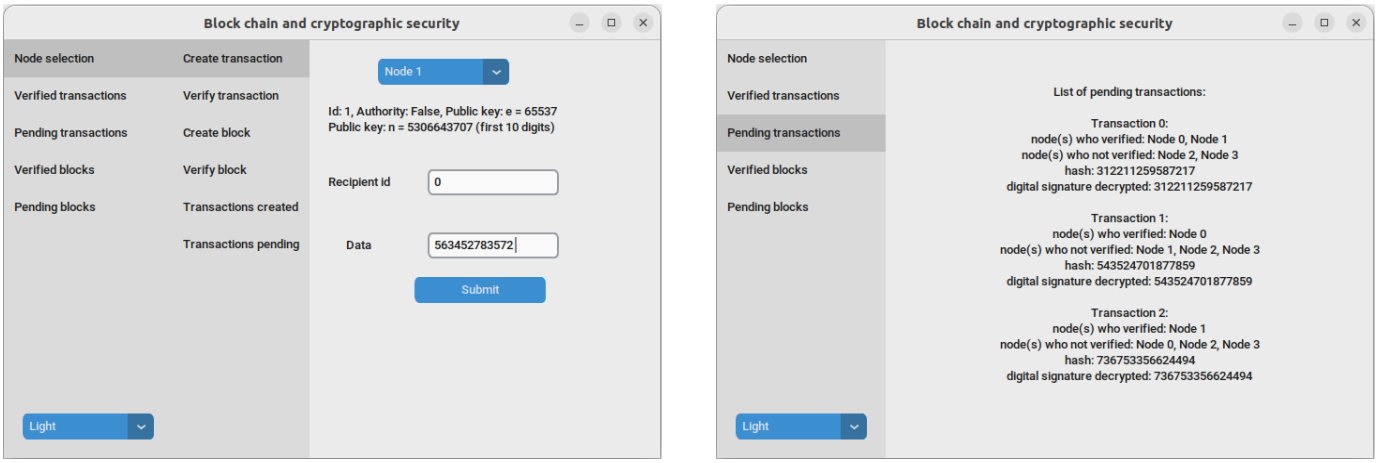

Fig. 4. Creating transaction

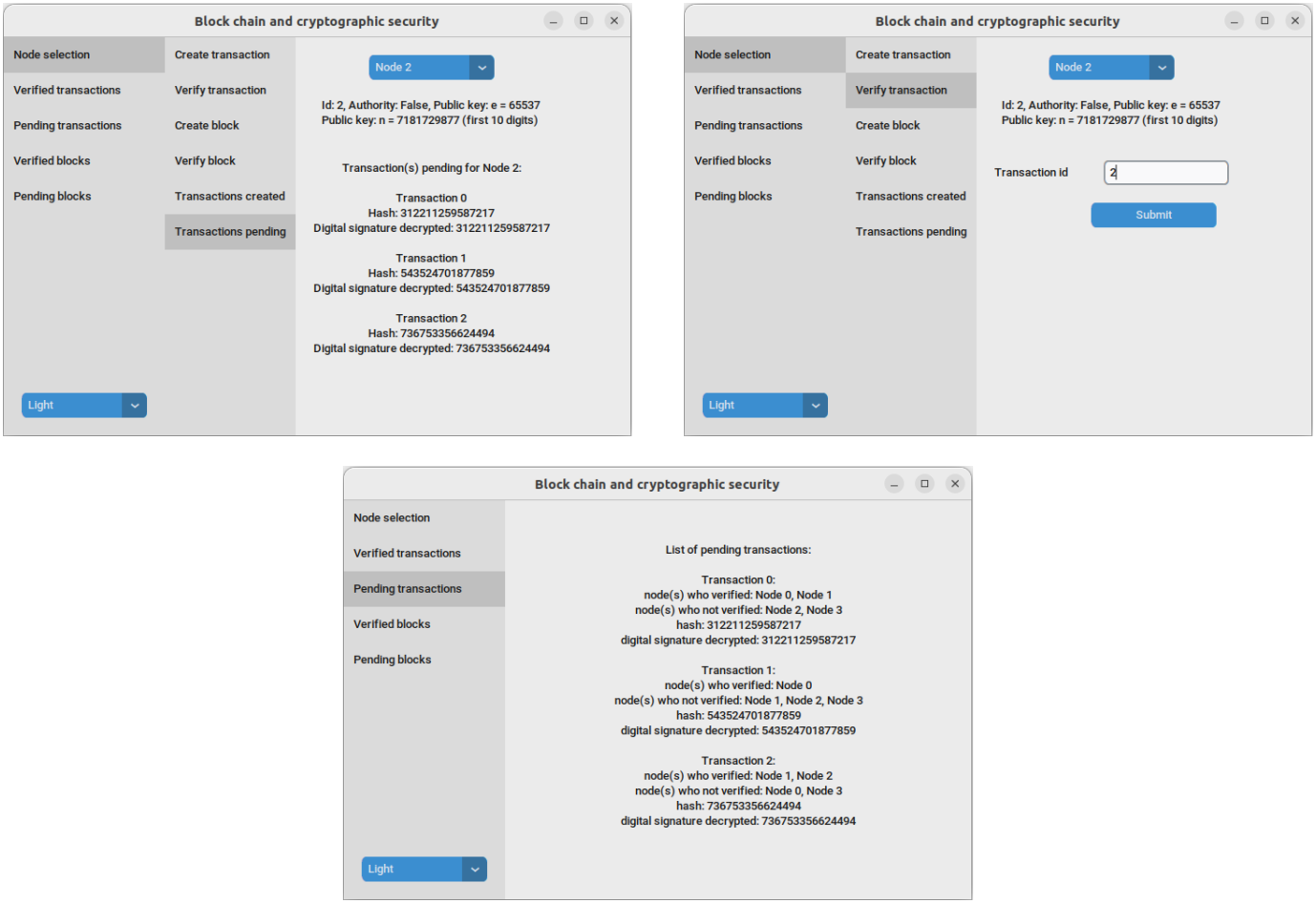

Fig. 5. Verifying transaction

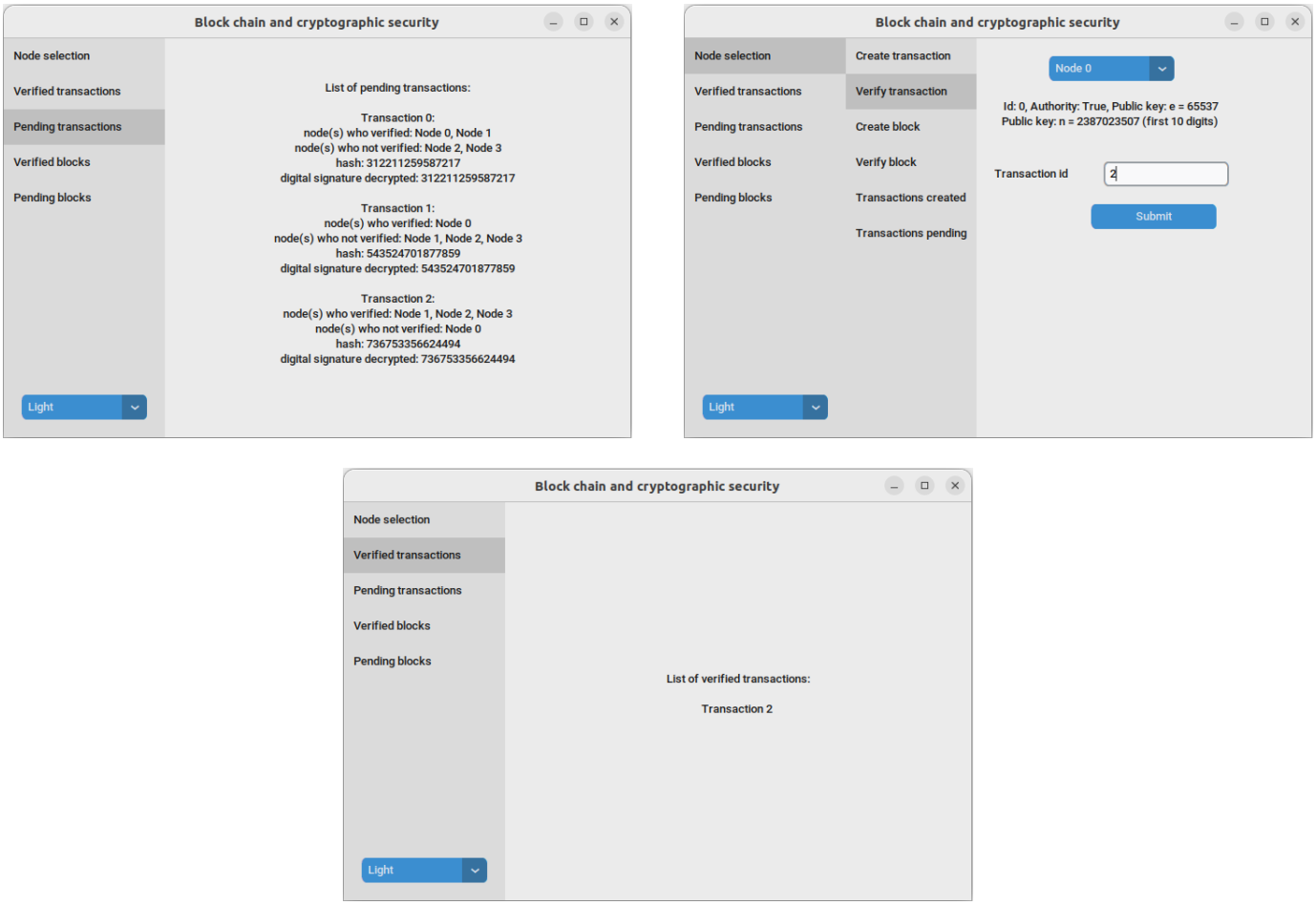

Fig. 6. Validating transaction

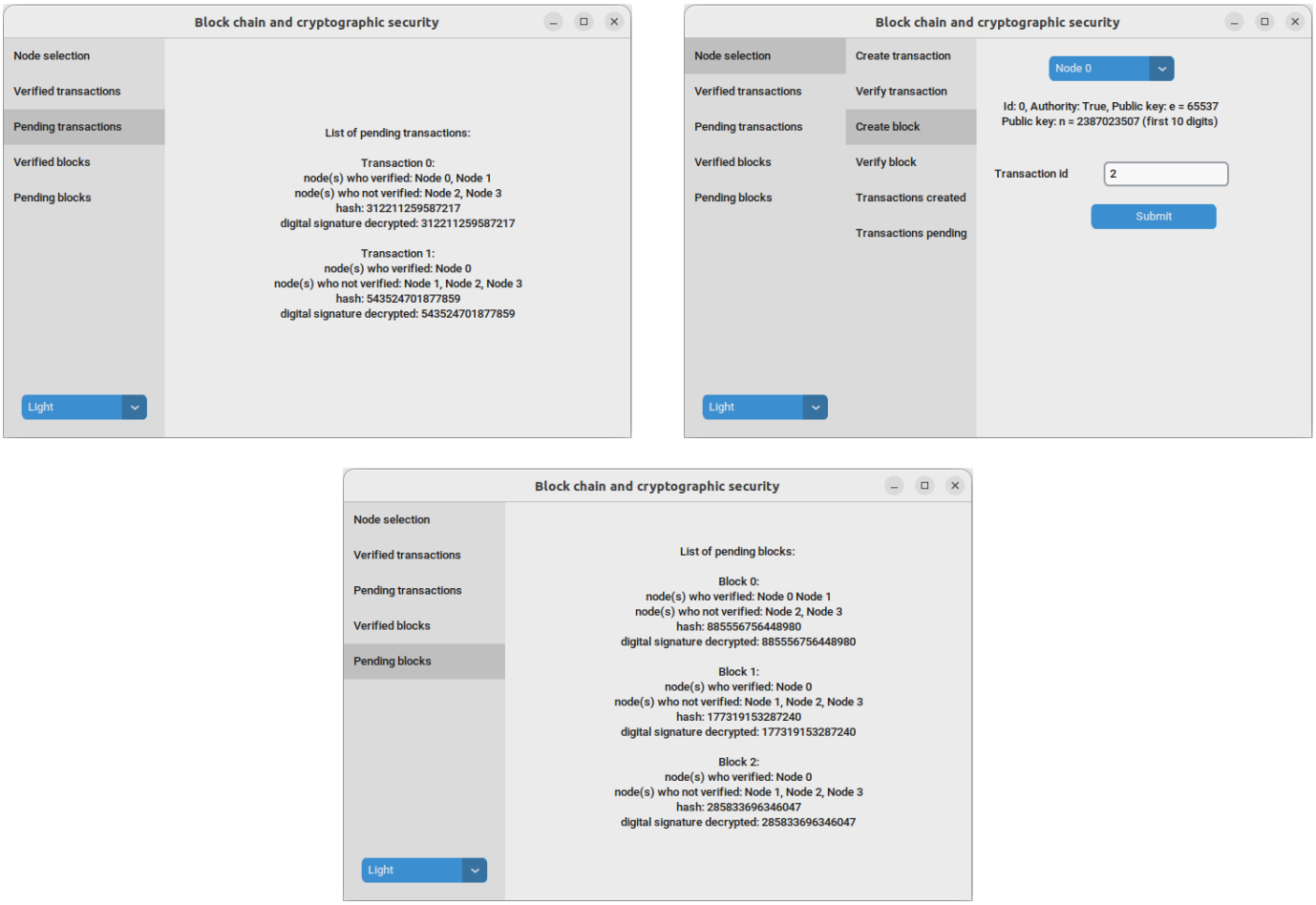

Fig. 7. Create block

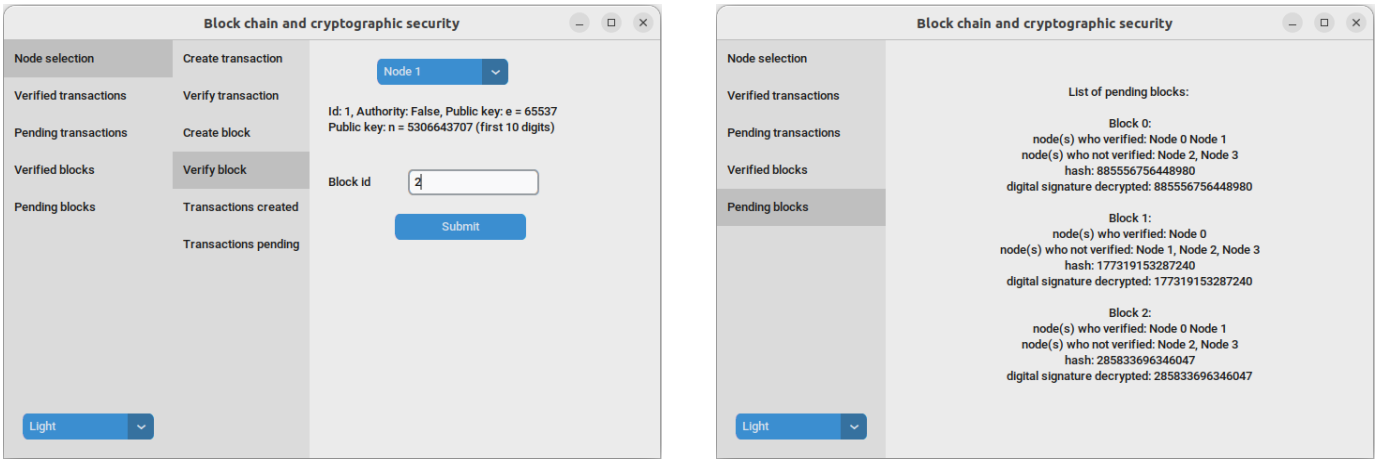

Fig. 8. Verifying block

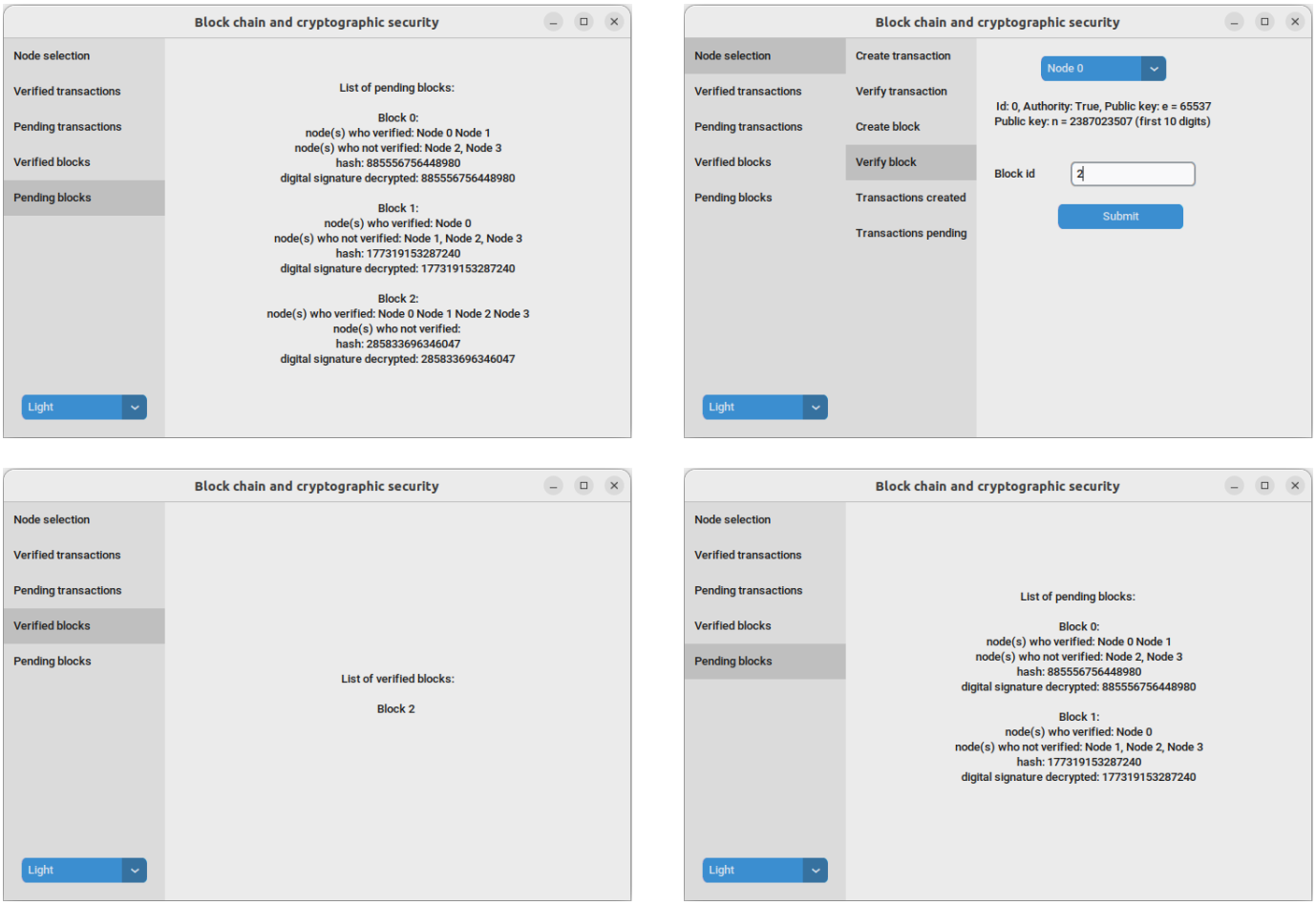

Fig. 9. Validating block

IV. Discussion

In this section, we will explain in detail the various use cases illustrated by the screenshots in the previous section.

A. GUI Menu 1

The first figure in the results section (see Fig. 3.), shows the first available menu (menu 1) presented earlier in the flow diagram (see Fig. 2.).

B. Creating transaction

In this scenario (see Fig. 4.), node 1 creates a transaction with node 0 as the recipient and 4563452783572 as the data. Transaction number 2 then appears in the list of pending transactions, with information about the nodes that have verified or not verified the transaction, the transaction hash and the decrypted digital signature. The digital signature ensures that the content of the transaction has not been altered since it was created.

C. Verifying transaction

Once transaction 2 has been created, it must be validated by all the nodes in the network. Here (see Fig. 5.), node 2 verifies transaction 2, checking that the hash corresponds to the decrypted digital signature. The verification of node 2 can be seen in the pending transactions tab, where node 2 has been added to the ”node(s) who verified” section. As nodes 0 and 3 have still not verified the transaction, the transaction cannot be validated by a network authority. This is the strength of the blockchain: it is the network’s unanimity that guarantees the authenticity of the transaction.

D. Validating transaction

Validation of the transaction is possible under two constraints: all the nodes in the network must verify the transaction and the node submitting the validation must be an authority. In this scenario (see Fig. 6.), the transaction is verified by all the nodes except node 0. Below the drop-down menu for selecting nodes is the information for the current node. In the case of node 0, we can see that it is an authority and can therefore validate transactions and create blocks if the conditions are satisfied. Transaction 2 is therefore verified by node 0, which at the same time validates it by ensuring that all the nodes in the network have verified the transaction. Once the transaction has been validated, it disappears from the ”pending transactions” tab and moves to the ”verified transactions” tab.

E. Creating block

Creating a block allows one or more transactions to be stored securely. To simplify matters, we have implemented a solution that takes only one transaction as input. The block can only be created if the transaction has been validated and the current node is an authority. These constraints are met in our case (see Fig. 7.) since transaction 2 is validated and node 0 is an authority. Once the operation has been completed, the new block can be found in the ”pending blocks” tab.

F. Verifying block

In the same way as transactions, blocks can be verified by network nodes. In our case (see Fig. 8.), node 1 verifies block 2 by ensuring that its hash matches the decrypted digital signature. This guarantees the integrity of the entire block.

G. Validating block

Once the block has been verified by all the nodes in the network, it can be validated by an authority. In our case (see Fig. 9.), node 0 validates the block by ensuring that all the nodes in the network have verified the block. In addition, the authority checks that the $previous_hash$ attribute of the block matches the $hash$ attribute of the previous block in the chain. In this way, node 0 ensures the integrity of the chain as a whole.

V. Conclusion

Blockchain is still a new technology, but it holds great promise. In the banking and medical sectors, this system is perfectly suited to securing sensitive content. The advantage of this method is that there is no need for a third party to ensure the integrity of a transaction. However, it is more resource-intensive because a network node has to perform calculations even when it is not involved in a transaction. As things stand, we can see the limit of the method when the number of nodes becomes very large. How many nodes are needed to consider a transaction valid? Who should check in priority?

Implementing this system has given us an in-depth understanding of how it works. The version delivered has all the features specified in the specifications, but there is still potential for improvement. For the moment, a block only contains a single transaction, which could be increased. The process of verifying and validating transactions and blocks could be multi-threaded in order to speed it up.

References

[1] Massimo Di Pierro. "What Is the Blockchain?" In: *Computing in Science Engineering* 19.5 (2017), pp. 92–95. DOI: 10.1109/MCSE.2017.3421554.

[2] D Rachmawati, J T Tarigan, and A B C Ginting. "A comparative study of Message Digest 5(MD5) and SHA256 algorithm". In: *Journal of Physics: Conference Series* 978.1 (Mar. 2018), p. 012116. DOI: 10.1088/1742-6596/978/1/012116. URL: https://dx.doi.org/10.1088/1742-6596/978/1/012116.