Pawn Goats: The Blockchain Pawn Shop

CSEC 559 Final Project

Nick Geigel, Alex Nasca, Trevor Radomski

Introduction

Traditional pawn shops involve interacting with a trusted business, but can often result in unexpected outcomes such as the customer's item being sold early, or the repayment options being changed. More importantly, if a customer wants to pawn an item, they are often required to give up large amounts of personal information before the transaction can begin. Our solution looks to effectively solve these problems through the use of the blockchain and smart contracts. By using the blockchain, we aim to increase integrity and availability in the pawning community.

Methodology

Our design methodology can be broken up into a few main pieces. Most critically, all business logic will be handled by smart contracts, such as order creation, payment, shipping confirmation, and information storage. This use of the blockchain to handle business logic ensures both the integrity and availability of all pawn (transaction) related information. These smart contracts will be programmed in Solidity and will be completely decentralized. Through the use of the blockchain, we eliminate the need for user authentication, therefore eliminating the need for centralized account creation or maintenance. Instead, a user performs all transaction actions using an order ID and makes all payments using their own Ethereum wallet.

We will then create centralized frontend and backend servers that will act as a way for the customer and business to interact with the smart contract and therefore perform the transaction. The frontend server will be created using Angular. This server is responsible for displaying information about the customer's transaction, taking in user input, and sending this input data to the backend server's API. The backend server will be created using NodeJS, ExpressJS, and Ethers. The backend API serves as a link between the centralized frontend and the smart contract. The API closely mirrors the smart contract's functions, with only a few routes involving custom logic.

To provide the customer with some flexibility in their transaction, the pawn shop also provides three predefined interest rate packages that the customer can choose from when creating their order. The repayment timeframe is directly correlated with the interest rate, and therefore the customer can decide how long they would like their loan to last.

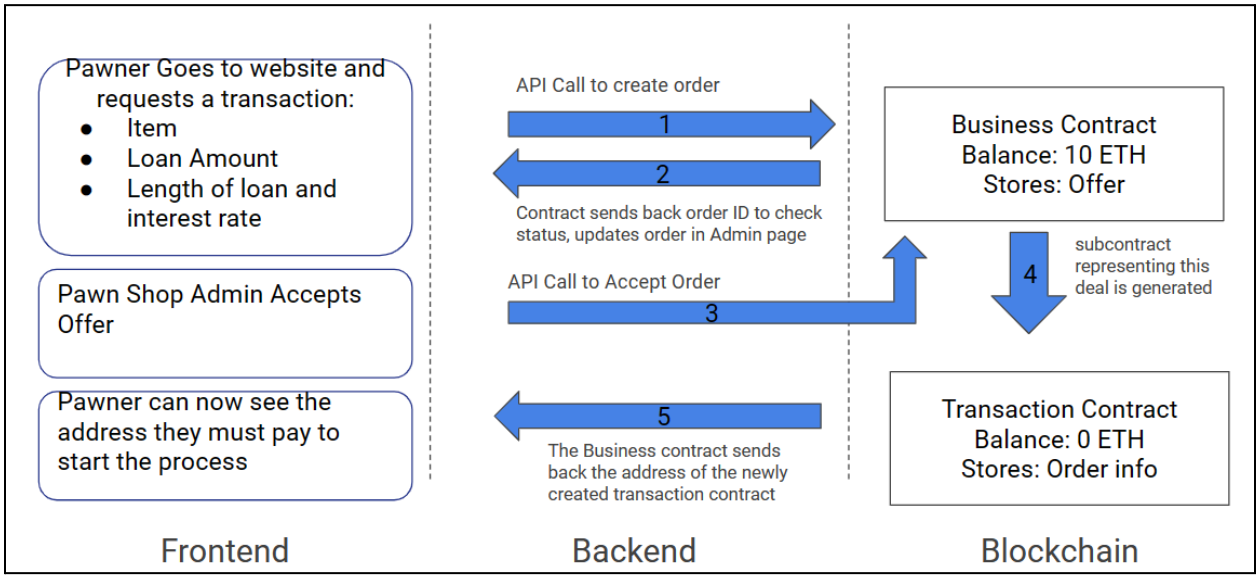
Results

Using the components described in the previous section, we were able to create an effective Proof of Concept (POC) system that replicates that of a real-life pawn shop. Most of our initial development time was spent planning how a transaction would play out from start to finish, and from there we created the necessary smart contract functions. Once all smart contract functions were implemented, the backend API was created, and the frontend could now interact with the smart contract via HTTP.

In order to evaluate the effectiveness of our pawning POC, it is crucial to first understand our devised system for how a transaction would play out. A customer begins by creating an order proposal. In doing so they provide the name of the item, a description, a desired price, and their selected interest rate. Upon receiving an order proposal, the pawn shop administrator (administrator) can then accept or deny the proposal. If the order is accepted, the customer is then required to submit a small stake before shipping their item. The purpose of this stake is to reduce the risk for the pawn shop in case the received item is not as described in the order proposal. After the shipping stake is received, the customer ships their item and submits a tracking number to the contract. To ensure confidentiality, only a hash of the tracking number is stored which could be compared to in the future. Once the pawn shop receives the item as described, the administrator confirms receipt of the item, and the shipping stake is returned to the customer. Simultaneously, the pawn loan is also dispersed to the customer, the loan-repayment timer is started, and the pawn shop submits a stake to the smart contract worth double the value of the loan. The purpose of this stake is to ensure that the pawn shop does not sell the customer's item early. If they do, the stake would be given to the customer as collateral.

At this point, the pawn shop waits for repayment from the customer. The repayment amount is calculated by the second using the interest rate. At any point the customer can request to view their repayment amount, at which point the amount is locked for 10 minutes, allowing the customer time to repay. If the customer does not repay within 10 minutes, they will need to request a new repayment amount. Once the customer repays, the pawn shop then ships back the item to the customer. Finally, once the customer receives their item, they confirm and the company's stake and interest made during the transaction is returned to the main contract wallet. At this point, the transaction is complete.

Delving further into the details of the blockchain, we developed two contracts within our Solidity file named “BusinessContract” and “TransactionContract”. The BusinessContract is only deployed one time and serves as the externally facing contract that will be accessed by the backend Ethers. Once an order proposal is created by a customer, and then accepted by an administrator a TransactionContract is created. These transaction contracts contain all relevant information and functionality to one specific transaction. The backend Ethers does not directly access these unique contracts but instead, they are accessed using a mapping and getter function within the BusinessContract. This TransactionContract serves as the payable address for which the user will stake to, repay, and receive funds. This design hierarchy allows us to efficiently access the smart contract using Ethers, as well as allowing for the easy management of transaction details and functionality, therefore reducing administration complexity.



[Figure 1: Initial order creation and TransactionContract initialization]

Discussion

The system described in the previous section is sufficient for a POC but does bring with it some concerns. In an effort to move towards zero trust, we utilized the idea of staking to force both parties to have some investment in the success of this transaction. Unfortunately, while investing the stake is easy, claiming the stake in the event of malfeasance has turned out to be significantly more challenging. More specifically, if the pawn shop were to sell the item early,

the customer would need some way to tell the smart contract that they never received their item back and they would like to claim the stake as their own. However, this is not an easy task and with the time available for this project we were not able to implement a solution.

Furthermore, while many of our interactions are digital, the system does rely on the postal system for transporting the customer's item. In the event that shipping goes wrong (the item is damaged, shipped to the wrong address, or lost), there should be some way to void the transaction, therefore returning the stake and loan to the customer. At this point, the customer would be forced to dispute the shipping problems using traditional channels.

It has also been suggested that the shipping stake from the user is counterproductive, since most pawn shop customers do not have any money to begin with, and therefore cannot afford a shipping stake before their loan is dispersed. However, we feel that the anonymity that is afforded to the customer by this system is a luxury, and therefore our target audience is not exactly the same as a traditional pawn shop.

Since our system's transaction data is available to the public, we need to be mindful of the information we store on the blockchain, such as hashing the shipping number. A positive outcome of making all transaction data public is that potential customers have the ability to review past transactions and compare prices for similar items. This affords the customer a level of transparency that a typical pawn shop does not provide.

Future Works

As described above, this digital pawning system is not without flaws. Our system acts as a proof of concept to demonstrate that a digital pawning system is possible using the blockchain. Still, future changes will be necessary to bring this system up to the quality that is required for production and real-world use. We foresee the stake claiming in the event that the transaction goes awry as the largest outstanding feature since this would directly correlate to financial security for both the customer and the pawn shop. Future development could utilize some sort of a timing mechanism to automatically release the funds to the customer if they do not receive their item before a certain time. However, this method may still require some degree of trust and therefore would require further research.

Similarly, further developments could be made in the interactions between the user and the pawn shop. This could include direct communication (chat messages), deal negotiations, and image exchange. For high-value transactions, it is critical that the pawn shop knows exactly what item they are receiving, and the inclusion of images may help aid in this.

Conclusion

The system we have created for this POC project effectively facilitates the anonymous pawning of items from a customer to the pawn shop through the use of the blockchain and smart contracts. This project facilitates transaction creation, loan dispersal, and interest repayments. This system was accomplished using a mixed structure of decentralized smart contracts, as well as centralized NodeJS-based frontend and backend servers. This system leaves room for future development, but overall serves as an effective POC for an anonymous digital pawn shop.