# BitTrade: A Pure Bitcoin Implemention of Smart Property

Karthik Dhore, Ben Stallworth, Keji Xu

Princeton University
COS 597E Final Project

**Abstract**

*We present an implementation of smart property in Node and Python, relying on the properties of the bitcoin to create a digital representation of property on the block chain. Following the structure of atomic transactions as presented by Mike Hearn on the Bitcoin Wiki, we implemented the protocol that allows two separate parties to partially sign the same transaction when agreed upon. We then used metadata in the transaction to store information about the property and the transaction, making ownership changes verifiable by anyone with access to the block chain. Finally, protocols were implemented for our server to communicate with the smart property on a web-based platform.*

## I. Introduction

Smart property is the concept of having the ownership of physical (or non-physical) property represented digitally in a decentralized manner. Bitcoin, a widely used decentralized cryptocurrency, which uses an append-only ledger to record transactions, can be used as a platform to control the ownership of smart property; using Bitcoin's block chain, addresses can be linked to physical property, such as a car, or non-physical property, such as shares in a company. Having a decentralized protocol for property ownership allows property to be traded and exchanged with reduced trust and potential for fraud.

Existing attempts at an implementation of smart property have either involved an addition to the Bitcoin protocol or the creation of altcoins to support these types of ownership structures. Colored coins, a bitcoin minting protocol by 'coloring' certain coins to represent a smart property, relies on the OP_Return operator to store metadata on the block chain. Altcoins, such as Ethereum, have forked from the bitcoin protocol to create protocols for more complicated financial functions and transactions. However, as it is difficult for altcoins to become as widely used as Bitcoin is, implementations of smart property is likely to be more successful if done on the Bitcoin block chain rather than on an altchain.

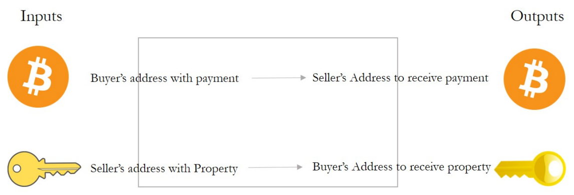
In our project, we abstract away all of the details of complicated protocols and present an intuitive implementation of smart property that allows a user to create an address on the Blockchain that is linked to the ownership of a piece of property (physical in this case), a protocol that allows users to transact the ownership of this property with minimal trust, and a protocol to claim and verify ownership of the property.

In the subsequent sections, we describe the theory to put a physical asset on the block chain and how ownership changes and verification should work in smart property protocols. We then describe our implementation of smart property transactions in Node, mainly using Bitpay's Node implementation of the Bitcoin protocol, called Bitcore. After implementing the transaction protocols, we then implemented the interface for the owner to claim ownership of the physical property, using a Raspberry Pi as a smart property who's ownership was controlled by an address on the block chain. Finally, we implemented a web interface for two parties to create smart properties, transact them, and finally use them.

## II. Theory

The owner of a piece of smart property also holds the private key of an address on the block chain that is linked to the smart property. Transactions involving both the property and the exchange of bitcoin as payment involve the simultaneous transaction of these two inputs, signed by different users, at the same time. This transaction, which is called an atomic transaction, involves the buyer of the asset, sending a partially signed transaction to the seller, who then signs their respective input (the one associated with the property) to complete the transaction.

**Figure 1:** *Illustration of Atomic Transaction with Smart Property*


Inputs
Outputs
Buyer's address with payment
Seller's Address to receive payment
Seller's address with Property
Buyer's Address to receive property

In more detail, the address associated with the smart property, upon its conception, has a token amount of bitcoin deposited on it (as an unspent output from a transaction). The piece of property has a protocol to communicate with the blockchain and an API for the owner to claim ownership and access it. When the owner of the property would like to sell it, they select a key to receive the payment, and the buyer generates a key for the new ownership address for the property. The buyer then creates a transaction with his payment (which he then signs) and the property's address as inputs, and the seller's address to receive the payment and the property's new address as outputs. The buyer than sends the partially signed transaction to the seller, who then signs the input associated with the property and then broadcasts the transaction. After the transaction is confirmed and there is enough confidence, the buyer can then present this transaction to the smart property, which then updates its ownership credential s(address and transaction) after verifying the validity of the transaction.

## III. Implementation

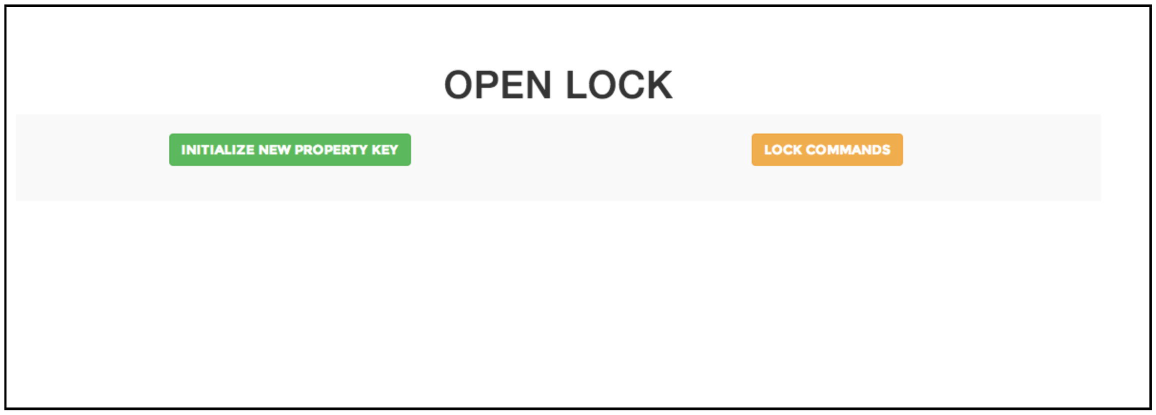
In our implementation of smart property, we create two major pieces of code: a "lock" module that runs on each smart property's hardware and a server for communicating with each property and conducting transaction. We capture all the essential aspects of smart property and transacting with it on the block chain. With our website, users can initialize a piece of smart property by creating an address that is associated with it, buyers and sellers can transact pieces of property in an online marketplace, and users can verify ownership of their smart property, in effect unlocking it for use. The server was written in Node.js, using Bitpay's javascript implementation of the bitcoin protocol, Bitcore.io. The code local to the smart property was written in Python, and provided an API for the sever to access and communicate with. Each piece of property runs its own Python module, instantiated with its respective parameters (address, transaction history) from the server. In this way, we have created a protocol which separates the smart property marketplace and atomic transactions from the actual piece of physical property.

## I. Initializing the Smart Property

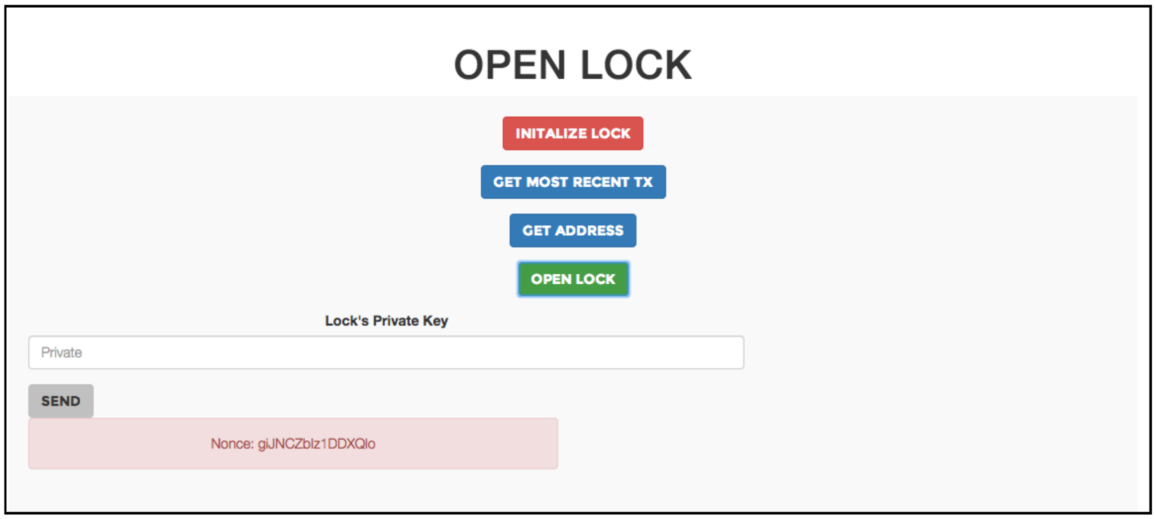
Creating an object that has a smart property first requires obtaining some hardware that is able to run Python and downloading our module to it. The user must also decide what the property will "open". That is, ownership of the smart property provides access to some entity, whether that is a piece of hardware, a program, or even a car. The user attaches the lock to this entity by filling in the template function provided in user.py. This code will be executed whenever the lock is opened. As our project focusses on the implementation of the smart property protocol, we leave it up to the user to make sure that the program placed inside the lock cannot be modified or run by other programs.

For our project, we use a Raspberry Pi as a piece of hardware who's ownership/access is controlled by ownership of the smart property. We wrote a module in Python to govern the interaction between the Raspberry Pi and the server. We modified the Raspberry Pi's initialization file to immediately start the python script on boot, which serves as a "lock" on the Raspberry Pi as it prevents from anyone from using it without passing the script's verification. Thus, the Raspberry Pi's usage is a property that is tied directly to the block chain and can be sold in a bitcoin transaction.

## II. Unlocking the Property



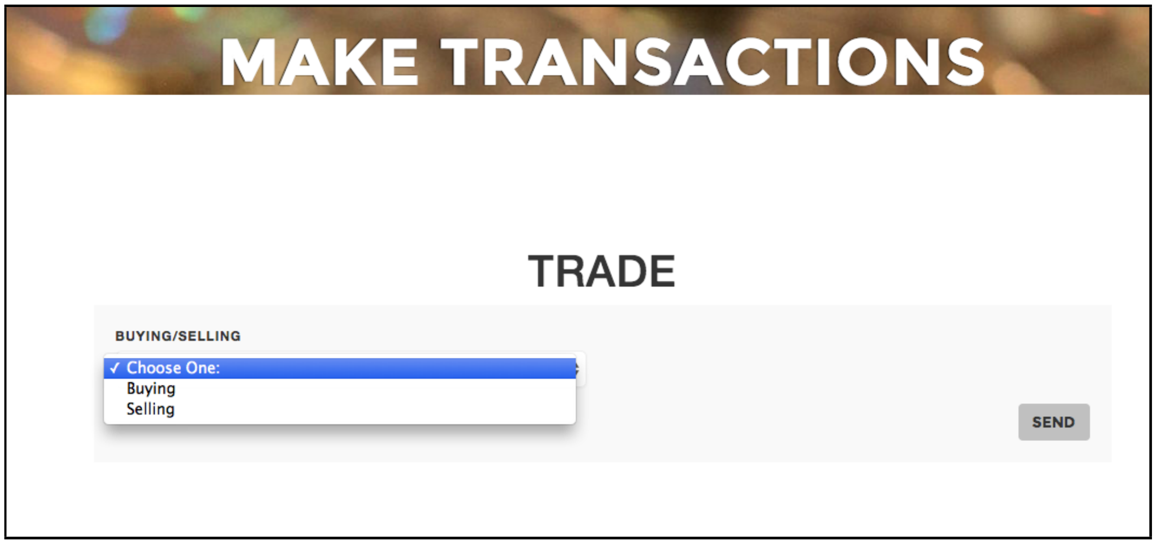
When the lock script is run (by running "python lock.py"), a python server is created on the hardware that first updates the locks ownership address and then waits to be contacted by our Node.js server. The lock updates its address by examining the blockchain and seeing if the unspent transaction output it has saved has been used in any transactions. It follows this chain of transactions until arriving at its current output, from which it gleans its new public address. The owner of the lock can then go to our website and enter in the hardware's IP address. Our Node.js server then uses the ZeroRPC library to contact the lock, which will produce a nonce and send it to the server. The server then asks the owner of the lock for their private key in order to sign nonce. Once this is obtained, the server proceeds in signing the nonce and sends it back to the lock. The lock verifies that the signature is correct by checking it with the public address it has saved in its log file. If the verification is correct, then the server closes and the script placed in the function "unlockedProtocol" in user.py is executed.



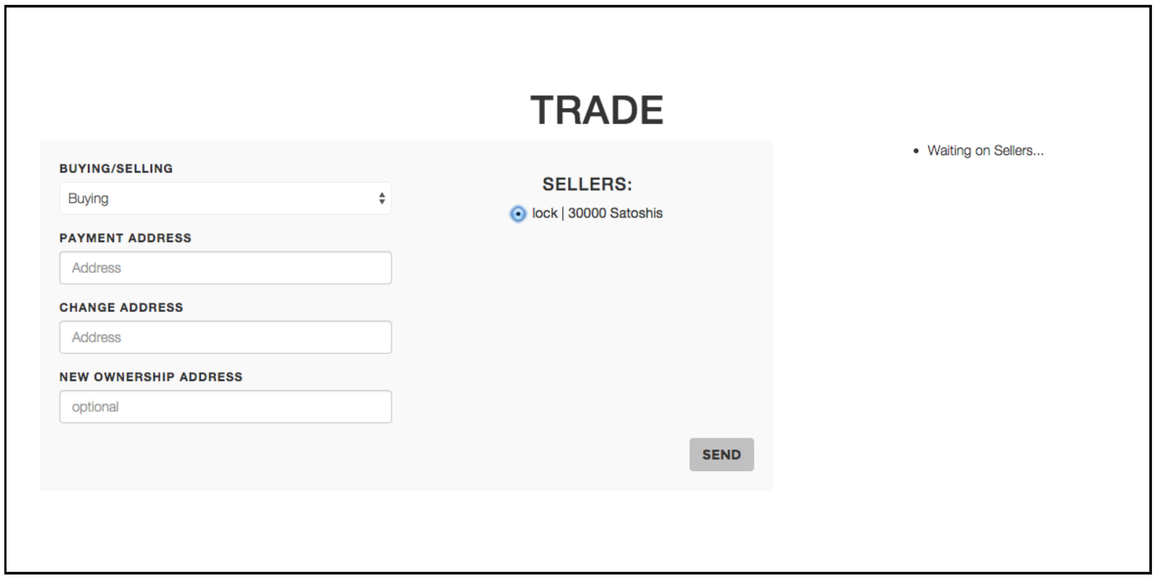
Nonce: giJNCZblz1DDXQlo

In our case, when the Pi turns on it executes the script immediately. Thus, the only way to gain access to anything on the Pi is to present the correct private key on the website. Upon verification with the server, the script simply exits (the user.py file is just template code) allowing the full use of the Raspberry Pi software.

## III. Buying and Selling



Critically, our website also allows for the transferring of ownership of the smart property. In order to sell a smart property, the seller goes to the website and lists the property with a description, a price, and a public address at which to receive payment. At the same time, a buyer can go the website and see a list of sellers. The buyer chooses the property they would like to buy and proceeds by entering the public address that contains the bitcoins they would like to pay with.



TRADE

BUYING/SELLING: Buying

PAYMENT ADDRESS: Address

CHANGE ADDRESS: Address

NEW OWNERSHIP ADDRESS: optional

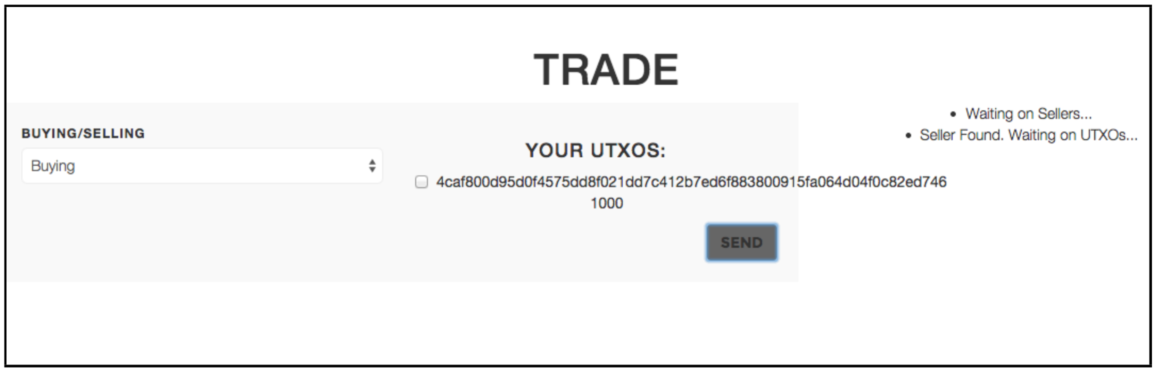
SELLERS:

[✓ lock | 30000 Satoshis]

SEND

- Waiting on Sellers...

The server then presents the buyer with a list of unspent transaction outputs related to that address (using the Blocktrail API), and the buyer picks the ones they would like to pay with. The buyer also inputs a change address for any extra bitcoins and finally signs the transaction by entering in their private key. This partially signed transaction is presented to the seller, who signs it with the private key they use to unlock the smart property. The server then broadcasts the transaction to the bitcoin network. We obtained a list of nodes to broadcast to from http://blockchain.info/connected-nodes. The server also presents the buyer with the new private key and public address associated with the smart property, which are generated by the server during the transaction's creation. As mentioned in the previous section, the lock updates automatically by considering the block chain every time it is run, so there is no need to present the lock with any evidence that it has been sold.



TRADE

BUYING/SELLING: Buying

YOUR UTXOS:

☐ 4caf800d95d0f4575dd8f021dd7c412b7ed6f883800915fa064d04f0c82ed746
1000

SEND

- Waiting on Sellers...
- Seller Found. Waiting on UTXOs...

Using Node.js as the critical server allows for easy asynchronous calls through Socket.io. During the buying and selling exchange, everything is happening in real-time. Thus, multiple buyers and multiple sellers can be on at the same time, pairing up and interacting fluidly.

## IV. Further Work

The study of Smart Properties is extremely recent. Thus, there are vast and varied opportunities for future development:

1. Offline Signatures

   One major potential security flaw with our procedure is that private keys are entered in to the server in order to sign transactions and prove ownership of the smart property. Ideally, these operations would be performed either in client-side javascript or in an application located directly on one's computer. Passing these keys over internet may and should make many users nervous, especially as eavesdropping and snooping attacks become more common. When using our website, we encourage users to create a new unspent output with only the money they need to spend on the transaction.

2. Offline Interaction with Lock

   The lock could also be implemented in an offline fashion, with ownership being updated by someone presenting the lock with the transaction containing the transfer of ownership and the block headers from the lock's current unspent output to that transaction. The advantages of this implementation is that no internet connection needs to be maintained for the lock. The disadvantage is that the user is forced to give the lock a great deal of information, and may not even know exactly how much the lock needs.

3. Wallet

   Our application also lends itself extremely well to a wallet application. Such a wrapper around our protocol could make the trading of smart properties much easier. In such a system, users wouldn't have to deal with remembering specific addresses or private keys, as these could be handled by the wallet. A wallet would also ideally implement both of the above advances, resulting in a truly complete implementation of the system.

4. Further Hardware Implementation

   Perhaps the most obvious development opportunity and the opportunity with the most potential variety is the advancement of hardware implementations of the property itself. Our implementation was a very basic software script that is not especially secure to attacks. The Raspberry Pi can be wiped and reprogrammed, stolen, or possibly even ssh'ed into while the lock is running to circumvent the lock. However, future hardware will be more secure, more creative, and more useful. If a fully implemented wallet app were to exist on mobile platforms, hardware producers would be extremely incentivized to produce hardware tradable on the block chain. With NFC hardware, these mobile phones could not only be responsible for trading properties, they could also control the updating and verifying ownership of them. The canonical example, given in the bitcoin wiki page on smart property, is that of a car that is sold by tapping phones with the salesperson, and unlocked and started by tapping the owners phone against the car. The technology needed to make this happen is already here, and we fully expect smart property to exist in some form in the upcoming years.

## References

[1] Hearn, Mike. "Smart Property". The Bitcoin WIki. Jan 2015. Web. <https://en.bitcoin.it/wiki/Smart_Property>