Atomic Cross-Chain Trading and hash locks in general work well to ensure atomic transactions on on-chain states. Can something similar for other stuff (say, merchandise) be done?

Consider the case of a Seller selling a Product to a Buyer.

Assumption:

- Product being traded is any digital or physical asset that can be secured against unauthorized use through a digital key (d

).

- Buyer knows hash of key (h(d)

)

- Seller's keys (k_{pub}

/k_{pri}

), Buyer's keys (v_{pub}

/v_{pri}

)

Proposal for a protocol:

1. Seller Deployment

: Seller will publish a new smart contract that includes: Price of Product (P_d

), h(d)

, and a nonce (ID

). Seller also deposits an amount \mathcal{E}_S

1. Buyer Initialization

: The Buyer must pay the price P_d

for the product and makes a deposit \mathcal{E}_{B}

.

1. Delivery

: The Seller sends enc_{v_{pub}}(enc_{k_{pri}}(d || ID))

, to the Buyer.

1. Accept/Reject Delivery

: Buyer notifies the smart contract of either acceptance or rejection of the delivery. In case of acceptance, Seller gets P_d

, and all deposits are returned to respective parties. In case of rejection, Buyer must provide evidence of malice on Seller's part. This happens in the next reconciliation step.

1. If Buyer disputed delivery, then it must provide enc_{k_{pri}}(d || ID)

to the smart contract. After decrypting and obtaining d

, smart contract hashes it and compares with h(d)

. In case of mismatch, the Buyer is refunded P_d

and \mathcal{E}_{B}

, and Seller's deposit is slashed.

Find a game-theoretic analysis, and full proposalhere

Note: Decryption in a smart contract is a strong assumption, but I guess something like TrueBit or incentivized oracles could

be worked out for that.