Cryptographic protocol for marketplace (work-in-progress)

CRYPTOGRAPHIC PROTOCOL:

It would be valuable for a trustless marketplace for exchange of goods and services to exist. Currently there is trust between buyer and seller and the proprietor during such a transaction with all parties at risk from a single centralised point of failure.

P2P setup for sale of a digital asset for bitcoin. Dumb nodes on network communicate by plaintext broadcast messages. Security maintained with public-key ECC cryptography and cryptographic hashes. Key security for data ensured by distributing the private encryption key over multiple nodes with Shamir secret sharing algorithm. Funds in transfer protected by multi-signature bitcoin transaction where private keys never meet, nor are revealed to the network. Redundancy built into both the SSS and multi signature aspects of protocol.

First attempt and relies upon some trust in a random node which leads the decryption process – possibility to integrate a modified Rochambeau protocol to remove this..

Broadcast transactions during example payment for release of an encryption key to a digital asset or data:

Seller:
CreateNewTransaction ->
<-NewTransactionResponse
StoreNewTransaction ->

Buyer:
BuyerCompleteTransaction->
<-AliveCompleteTransaction
BeginDecryptTransaction->
<-ContinueDecryptTransaction
MultisignatureSignTransaction->
<-MultisignatureSignedTransaction
CompletedTransaction ->

In detail..

Sell protocol:

Client(sell):
Random hash C created: C = sha256(nonce, n)
New ECSDA key pair generation
Encrypt file/data with private key.
-> CreateNewTransaction broadcast to network **{**C,nonce,client public key**}**

Every node:
Upon receipt of new transaction request generates a new ECDSA key pair and a chained series of hashes originating from a random number. A response is broadcast and includes the last hash, F. Also sent is a signature of C made with the private ECDSA key and a bitcoin address derived from the public key. The signature is ECC encrypted with the client public key to prevent the public key leaking onto the network.
E = hash(B)
D = hash(E)
F = hash(D)
-> NewTransactionResponse {C,nonce, F, ECC client public key(node ECDSA signature of C),node bitcoin address**}**

Client(sell):
The client receives back responses and after a minimum time t and responses >=3 continues:
10 responses are randomly selected
The signature of C is revealed by ECC decryption with client private key.
The node public key is derived from C (message) and the signature.
Public keys are ordered in a reproducible fashion and an m-of-n multi signature script is generated.
A multi-signature bitcoin address is derived and likely fee calculated.

- compose raw tx for sellers price to sellers bitcoin address and node addresses for fee (1% price / n).

The multi-signature script is ECC encrypted with client private key.
Private key then converted into shares using Shamir Secret Sharing scheme (k-of-n) and inserted into an array with node ECDSA signatures for each corresponding node public key involved in multi-signature script.
Both secret share and node ECDSA signatures are then ECC encrypted with node public key (which only client knows).
-> StoreNewTransaction broadcast to network
{C, nonce, reveal n, multi-signature bitcoin address, btc price, ECC encrypted multi-signature script + seller bitcoin address, multi-sig raw tx, array containing [ECC encrypted with node public key (node ECDSA signature of C, private key SSS share, node bitcoin address), F**}**

Transaction broadcast to p2p layer, file/asset uploaded to network, published to dynamic web page. Buyer turns up and queries network for transaction details.

Buy protocol:
Client (purchase):
New ECDSA key pair generation. Signature of C with private key. Public key converted to a bitcoin address. Funds deposited by client into new address. Bitcoin moved from newly generated bitcoin address to the multi-signature bitcoin address which reveals the public key to the network. After network confirmation buyer broadcasts to the network. (This is a rate limiting step)

->BuyerCompleteTransaction
{C, ECDSA signature C**}**

Every node:
Nodes check BuyerCompleteTransaction to see if C matches a previous StoreNewTransaction record and if it does performs a checklist:

- for C: does multi-signature balance >= btc_balance?
- for C: does F match local node copy?
- does associated encrypted entry in StoreTransaction decrypt to reveal the signature from node private key (every node has generated submitted a public key but is this node a key holder?)
- Derive buyer public key from signature and message©.Check derived public key against input transaction public key for btc_price to multi-signature bitcoin address.If match then we have located the buyer and we have confirmed they hold the private key to that address.

If all criteria met then we have found a valid buyer who has paid and can cryptographically prove they sent the funds, whilst our node is holding an active key in this transaction.
New ECDSA key pair generation (key pair 2).
->AliveForCompleteTransaction
{C, F, reveal D, node public key 2**}**

All nodes which carry an active public key:
All active nodes which are involved in transaction reply using the reveal to prove they are legitimate. All active nodes verify hashes with reveal and fill array to see what m of n are online
If not 1st active response in list order then wait 10s for BeginDecryptTransaction response from 1st active node. If no such response arrives then error procedure.

1st active node in list:
if list completely populated or t elapsed:
decrypt shamir secret share with node private key 1
->BeginDecryptTransaction
{C, F, D, reveal E, node public key 2**}**

All remaining **active nodes**:
Upon receipt of BeginDecryptTransaction challenge active nodes must verify reveal E -> D -> F to ensure they respond with details of private key share to the correct node and with working encryption key.
paranoid repeat previous checks and balances..
If hash valid -> decrypt shamir secret share with node private key 1
ECC encrypt shamir secret share with 1st node public key 2
->ContinueDecryptTransaction
{C, F, D, reveal E, ECC encrypted with 1st node public key 2 (share)**}**

1st node:

- Populates list in order of encrypted shamir secret shares and ECC decrypts with 1st node private key 2
- Performs Shamir Secret Sharing recover(share) to reassemble the initial Seller private key.
- Decrypts the multi-signature script
- Private key ECC encrypted with Buyer public key
- Multisign for multi-sig raw tx + script with node private key 1
- New array for each node that verified active [F, ECC encrypted with node public key 2 (multi-signature script)]

After reassembling the private key, decrypting the multi-sig script and encrypting the private key such that only the buyer may decrypt it, it is time to perform the multi signature transaction. This involves sending each active node the script to sign.

->MultisignatureSignTransaction

{C, [F, ECC encrypted node public key 2(multi-signature script)**}**

All remaining **active nodes**:

Each active node after 1st decrypts multi-signature script and performs multisign on script and raw tx with node private key 1. Signature then sent back to 1st node ECC encrypted with 1st node public key 2. B revealed to provide further authenticity.

->MultisignatureSignedTransaction

{C, F, reveal B, ECC 1st node public key (signature)**}**

1st node:

- 1st node performs verification for each reveal B (-> E -> D -> F)
- ECC decrypts each multi signature in array with 1st node private key 2.
- Apply multisignatures performed and push tx to bitcoin network.
- After confirmation of transaction to sellers bitcoin address, private key ECC encrypted with buyer public key is broadcast allowing buyer to download and decrypt the file..

->CompletedTransaction

{C, ECC encrypted buyer public key (seller private key)**}**