Chainsmiths

Technical Architecture:
Re-Vault – a Bitcoin Vault
March 2020, Final Document (based on Draft 3e)

Foreword

This architecture is produced by Chainsmiths and commissioned by NOIA, a trading fund with bitcoin holdings. The aim is to let NOIA be their own custodian without having to rely on any third party. Most companies with significant bitcoin holdings currently use multi-signature schemes, NOIA hired Chainsmiths with the goal to improve the state of the art and increase the security model scope.

Not dealing with a third party custodian concentrates the risks to the company and its stakeholders. This architecture includes a significant amount of comments on where the new threats are and how to mitigate them.

While this architecture is designed to fit NOIA's specific business model, it will be used as a template for various vault implementations. Chainsmiths expects other businesses to use this model and hopefully contribute through code, audits and other improvements.

Audience

This document is aimed at people familiar with Bitcoin and key management. Understanding of the Bitcoin Script language is a plus but not a necessity.

This architecture is novel, Chainsmiths strongly recommends any end-user to be familiar with this document, especially around the risks and key management.

Chainsmiths interaction with NOIA is not limited to this document and this document should not be considered enough to mitigate risks or be "secure". Proper key generation, code review, full node infrastructure, key recovery scenarios, key-people personal security management, key storage and more are all part of a sane custody implementation.

Authors

The research, design and redaction of this architecture and document was performed by Kevin Loaec of Chainsmiths. Third party reviews were performed by Bitcoin Core contributor [name edited, anonymous], independent developer Alekos Filini, and Antoine Poinsot from Leonod.

A proof of concept is being developed by Antoine, the work-in-progress is available at:

https://github.com/re-vault/re-vault

Chainsmiths is a "blockchain" consulting firm established in 2015 and headquartered in Dublin, Ireland, with an office in Lisbon, Portugal. It offers unbiased advice, strategy, audits, architecture and technical due-diligence to clients ranging from global companies to startups and governments around the world.

Kevin Loaec is Chainsmiths' managing director since 2015. He previously co-founded a bitcoin hardware cold storage company, co-organized Breaking Bitcoin and Building on Bitcoin, launched The Block Lisboa, and supported many initiatives in the startup and fintech ecosystems around Europe. He specializes in security and strategy around Bitcoin and privacy.

Alekos Filini is currently an independent Bitcoin developer and researcher. He previously worked on the first drafts of the RGB Protocol at BlockchainLab and later joined Blockstream as a Software Engineer. Having worked on Liquid and the Green wallet, he has experience with Bitcoin wallets, complex multi-sig protocols, key management and HSMs.

Antoine Poinsot is an open source Bitcoin developer currently focused on the Lightning Network and interested in all aspects of Bitcoin.
He co-founded Leonod to promote Bitcoin technologies in France, through which he currently provides Bitcoin development services.

NOIA Capital

NOIA Capital is a Luxembourg-based hedge fund specialized in digital assets and blockchain technology. NOIA was co-founded by a team of four individuals based in Switzerland, Belgium and Dubai with a strong experience in digital assets, information technology, trading and venture capital. They created the fund to manage their own capital and are backed by a renowned Belgian family office. They currently have launched the fund for further investor participation. NOIA actively manages a liquid portfolio of digital assets with a long / short strategy and reserve a small percentage of their funds for selective direct equity investments.

Vocabulary and definitions

This section regroups the important vocabulary and concepts used in this document.

Bitcoin (units: bitcoins): monetary protocol independent of nation-states, corporations or any single individual. It includes both the monetary policy itself and the payment network.

UTxO: Unspent Transaction Output. Bitcoins available to be spent by whoever can fulfill the conditions indicated in its script.

Script (language): Low level, stack based programming language used in Bitcoin to define locking conditions in UTxOs.

OpCode: predefined function in Bitcoin Script.

OP_CheckSequenceVerify (OP_CSV): OpCode requiring a delay (in blocks or time) between when the transaction containing the UTxO is mined (*~processed by the network*) and when it can be spent.

Public key: in public-key cryptography, a form of identity. Here used for signatures.

Private key: in public-key cryptography, the secret corresponding to the public key. Here used to sign Bitcoin transactions.

Signature: cryptographic function to prove ownership of the private key corresponding to a specific public key, therefore also proving the origin of the message being signed. Here used to fulfill UTxO signature requirements in Bitcoin transactions.

Multi-signature transaction (Multisig): Bitcoin transaction spending from a UTxO with more than one public key in the locking script.

Pre-signed transaction: a Bitcoin transaction signed but not yet broadcast to the network.

Partially Signed Bitcoin Transaction (PSBT): Bitcoin transaction signed by some of the required keys, but not yet finalized nor ready to be broadcast.

Descriptor: standardized, high level representation of the spending policies of a wallet. Allows to move a wallet across different software stacks without having to re-implement the spending logic multiple times.

Specific to this document:

Locked funds: in this document, funds impossible to spend, either temporarily or permanently.

Stakeholder: in this document, one of the key holders.

Trader: in this document, one of the stakeholder allowed to (*partially*) sign spending transactions.

Emergency Deep Vault (EDV): an address to never use in normal circumstances, it exists as a way to move funds outside of the day-to-day Vault architecture in case of threat. Its keys have been generated and stored separately from the regular Vault keys, they must be hard to access and kept very securely.

Vaults (V_x): Receive-addresses for the fund. While address reuse is not an issue here, the design tries to generate a V_{n+1} after a V_n has received funds.

Unvaulting Transaction (UnVTx): pre-signed transaction allowing spending a (single) Vault UTxO while enforcing a revocation delay. Requires 4 signatures.

Spending Transaction (SpendTx): regular day-to-day transaction performed by the traders when spending from a Vault (technically, from a UnVTx). Requires 2 out of 3 signatures (among two traders and one of the other stakeholders), plus a co-signing server signature and a delay.

Cancelling Transaction (CancelTx): a double-spending transaction to prevent a SpendTx from being confirmed, while the delay is still active. Sends back to a Vault. Pre-signed transaction.

Emergency Transaction (EmergencyTx): Pre-signed transaction to broadcast only when a threat is detected. Spends a Vault UTxO or UnVTx Output to the EDV.

Cosigning server: an automated signing participant for the SpendTx.

Requirements

This specific architecture/document is designed with the following requirements in mind:

- 4 stakeholders in total
- Two people (Traders) doing the day-to-day movements of funds
- A business continuity option when one of the trader is unavailable
- A "cancel transaction" trigger to be used in case of mistake, valid for a few minutes to hours after funds are spent by the traders
- A flexible design to be hardened in case of big deposits (per UTxO, does not mitigate multiple small UTxO theft at the Bitcoin script level)
- A "move to Emergency Deep Vault" trigger in case of suspected threat on the architecture or the key stakeholders
- A bypass of the protections possible if all 4 stakeholders sign transactions, for example in need to move funds within the next block, unless sent to the Emergency Deep Vault

The following architecture requires each of the 4 stakeholders to pre-sign transactions every time new funds are deposited in a Vault, although not necessarily at the time of deposit nor together. This means if a single stakeholder stops signing, the funds will be locked. Mitigation measures are discussed in a further section of this document.

Threat Model

The threat covered under this architecture is:

- up to *n-1* (here *n=4*) participants/keys have been compromised. This includes key theft (malware for example), collusion of employees to steal funds, duress or a combination of these.
  From a technical perspective, only the signer should have access to their private key(s). Even in the event of a collusion, it is simple to claim it was duress or theft but near impossible to prove. These situations are therefore covered as equal/equivalent in this architecture.

As stated in the previous requirements section, only 2 people are required to sign for normal business operations. A 4 of 4 multisig does not fit this requirement.

This model DOES NOT cover:

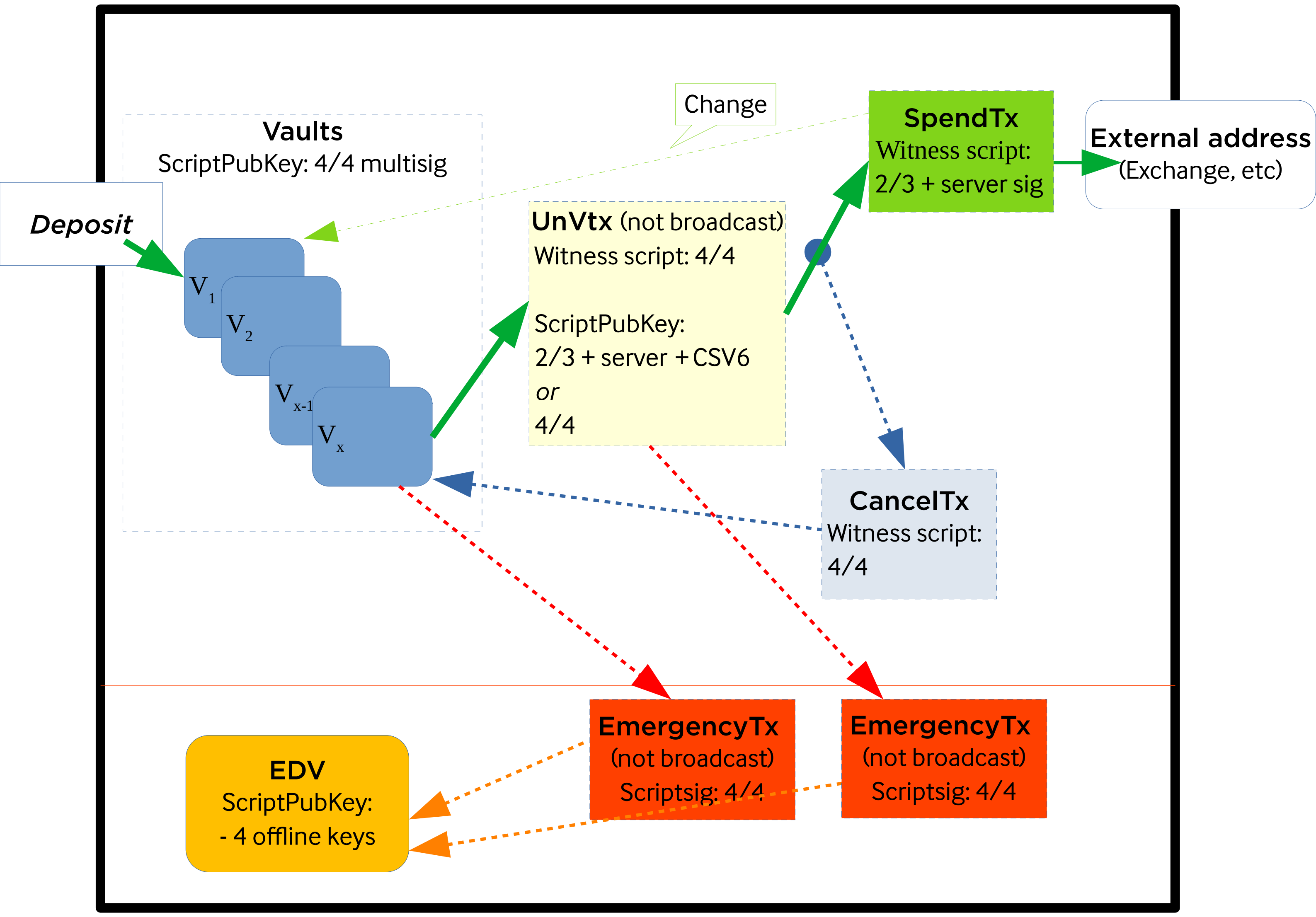
- a single participant **refusing to sign** new deposits transactions, therefore locking them. They could ask for a ransom for example, or prevent all business operations (DOS).
  We consider this to be resolvable by the traditional justice system, as the signers are identified.
- a single participant **permanently deleting their key and all backups**. New deposits (since the signing stopped) would be permanently locked. Such an attack is easily detected.
  This situation has no recourse under the current design, the new deposits are lost. It is possible to add redundancy, making the system resilient to 1 or more lost keys, but doing so also opens new vectors of attack and break the threat model. We leave the decision to the client but discourage implementing a fail-over.

The main compromise under this scheme is choosing to make it **near impossible to steal funds, at the cost of having a low protection against a single participant locking the funds.**

If we add the redundancies to make the 1 participant DOS attack impossible, we open the risk of a n-1 (*here 3*) people collusion to steal the funds (*which is the only solution to bypass a non-signer*).

While we need to have a key recovery process in case one of the participant disappear, we do not have a process that prevent key destruction.

Architecture



Rationale

Each deposit the fund receives should be protected under the present architecture. Deposit addresses MUST therefore all be Vaults with strict spending conditions and not spendable by any minority stakeholder without oversight. This must not impact the business operations too negatively, so the default behavior of the architecture is to allow spending with only the 2 signatures of the traders (or 1 trader + another stakeholder in rare case), while giving a cancellation or revocation window to any of the 4 stakeholder, plus any number of "watchtowers". This compromise lets the traders perform as usual even if the other 2 stakeholders are not available to review the transaction (approved by default).

All 4 stakeholders need to pre-approve (pre-sign) the spending of new deposits, i.e. the "approved by default" behavior only applies to deposits that have been previously acknowledged. While this somewhat limits the business operations, it prevents the spending of deposits unknown to the stakeholders.

An additional level of security, the Emergency Deep Vault, is added in case of a compromise of the fund or its key persons. The EDV is only spendable by 4 offline keys, not used in normal operations.

Addresses and Transactions

All transactions are considered to be SegWit to prevent malleability.

- **EDV**: Generated with 4 keys that will never be used elsewhere, and stored securely, not easy to access.
    - ScriptPubKey is a 4/4 multisig.
      4 <OFFLINE_1> <OFFLINE_2> <OFFLINE_3> <OFFLINE_4> 4 OP_CHECKMULTISIG

      NO REDUNDANCY IN CURRENT DESIGN: all 4 keys needed. Mitigation discussed in a further section

      Previous drafts of this architecture included a OP_CSV of two weeks. It was removed to ensure compatibility with most bitcoin software, not implementation-specific. We assume the 4 keys will be kept in hard to access and secure locations, enforcing a delay by itself.
- **Vaults**:
    - ScriptPubKey is a 4/4 multisig.

      4 <Trader1> <Trader2> <KeyC> <KeyD> 4 OP_CHECKMULTISIG

Different addresses can be generated by deriving the xPub of the 4 stakeholders at the same derivation path. Third parties MAY generate new vault addresses without the need for any stakeholder. (i.e. exchange implementing a protection on withdrawal addresses)

While the script is deterministic, **it is preferable to share new addresses directly** rather than giving the xPubs to third parties. Any of the stakeholder can generate new Vault addresses deterministically.

- **UnVTx**:
  - Pre-signed 4/4 (MUST sign the two EmergencyTxs and the CancelTx before signing this!)
  - Outputs scripts:

    2/3 (among the 2 traders and stakeholder C) multisig AND serversig AND OP_CSV 6 (*SpendTx*)

    or

    4/4 multisig (*CancelTx or EmergencyTx*)

    ```
    <Trader1> OP_CHECKSIG OP_SWAP <Trader2> OP_CHECKSIG OP_ADD OP_SWAP
    <KeyC> OP_CHECKSIG OP_ADD  OP_DUP 3 OP_EQUAL
    OP_IF
    OP_SWAP <KeyD> OP_CHECKSIGVERIFY
    OP_ELSE
    2 OP_EQUALVERIFY <SERVER> OP_CHECKSIGVERIFY 6
    OP_CHECKSEQUENCEVERIFY
    OP_ENDIF
    ```
- **SpendTx**:
  - Co-signing server will only sign each UTxO once, preventing a double spending at the time of OP_CSV expiration. The server will not sign if an UnVTx was already mined (CSV abuse)
  - MUST broadcast UnVTx input at the same time (*current Bitcoin Core policy prevents this, therefore we need to share it to the other stakeholders through a different communication channel*) to enforce OP_CSV full delay
  - If UnVTx is broadcast without its child SpendTx, CancelTx SHOULD be pushed by any participant or watchtower.
- **CancelTx**:
  - Pre-signed 4/4
  - Can double spend a SpendTx if broadcast within the OP_CSV delay
  - Sends the funds back to a Vault

- **EmergencyTx**:
  - Pre-signed 4/4
  - Spends from a Vault or from a UnVTx output
  - Funds go to EDV
  - SHOULD NOT BE USED under normal circumstances.

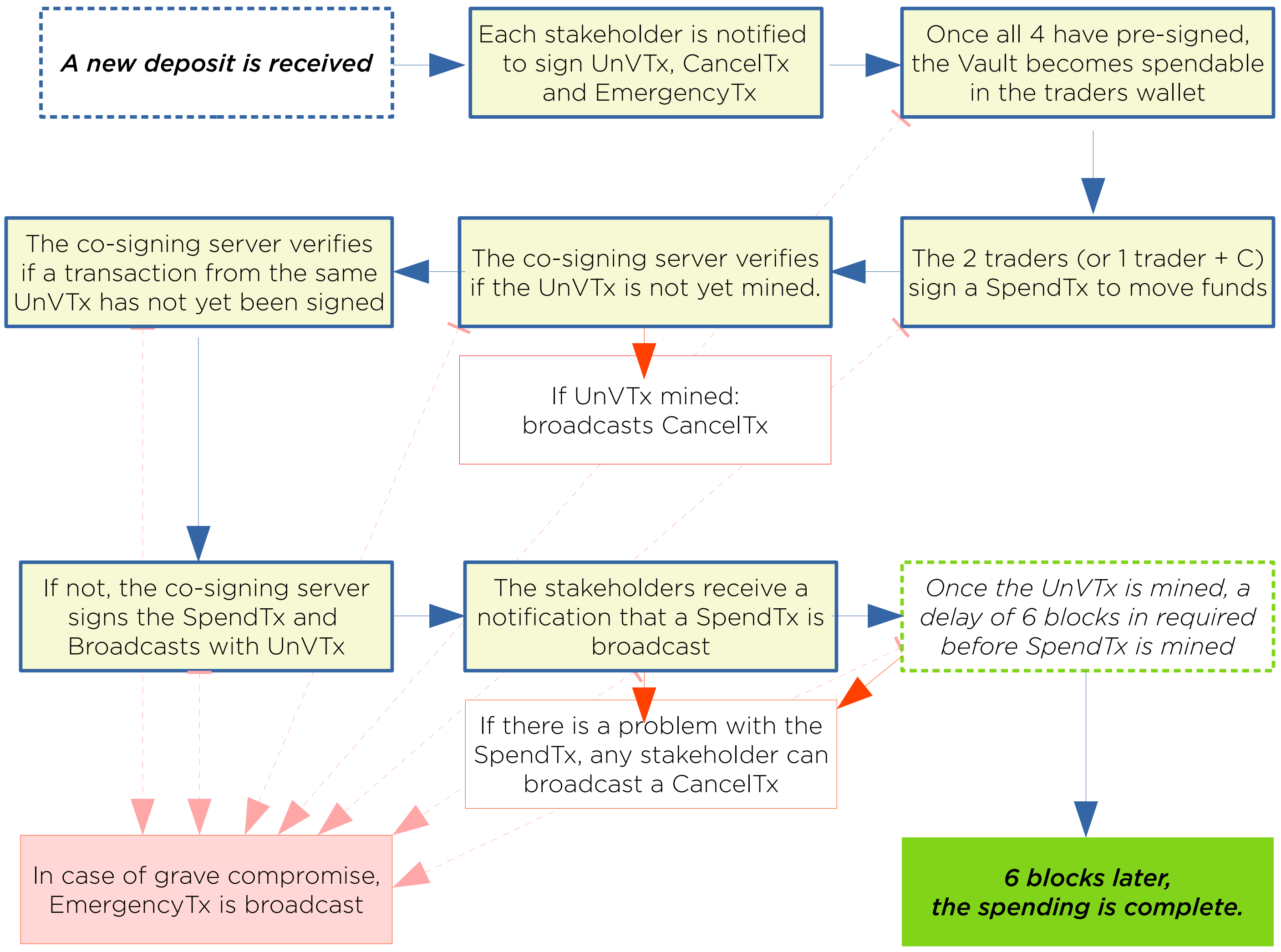
Process Flow

1. **Deposit**:
   Funds entering custody of the fund MUST be deposited to a Vault address. Vault address MAY be reused, new addresses can be generated by any of the stakeholders by deriving the extended public keys of all stakeholders at the same derivation path.
2. **Pre-signing**:
   Before a new deposit can be spent from a Vault, the stakeholders MUST all pre-sign four transactions and MUST NOT broadcast them to the Bitcoin network unless necessary. These four transactions MUST be signed in the following order:
   1. the EmergencyTx spending from the UnVTx
   2. the CancelTx
   3. the EmergencyTx spending from the Vault
   4. the UnVTx
3. (OPTIONAL) **Urgent Spending**:
   In the unlikely event of an urgent spending of Vaulted or UnVaulted funds, it is possible to bypass the time restrictions and co-signing server by having a transaction signed by all the stakeholders. This SHOULD NOT be used unless absolutely necessary.
4. **Watch**:
   The stakeholders, or watchtowers on their behalf, SHOULD be watching for spending transactions and be ready to broadcast a Cancelling Transaction if necessary. At least one of the 2 non-signer MUST be watching for a cancelling to go through. Automated watchtowers MAY be performing basic checks such as destination addresses, amounts, presence of UnVTx without a child.
5. **Spending**:
   The two traders can spend any Vault UTxO by broadcasting both the pre-signed UnVTx and the new SpendTx. *With the current Bitcoin Core policy, the SpendTx cannot be broadcast until the OP_CSV expires so it needs to be shared with the stakeholders through another channel.* Stakeholder C is added as a backup to this multisig, in case a Trader isn't available. The signature of a co-signing server is also necessary, its role is to prevent a double

spending attack from the traders at the exact block when CSV expires. In the presence of a change output, it SHOULD be sent to a Vault, unless very small.

6. (OPTIONAL) **Cancelling a transaction**:
   Any stakeholder, including the two traders, MAY broadcast a pre-signed CancelTx while a UnVTx has been broadcast and the SpendTx has not been mined yet (usually, 6 blocks interval, customizable). Doing so will result with near certainty to a re-vaulting of the amount, making the SpendTx invalid. This option SHOULD be used if anything is unclear about the SpendTx and MUST be used if an UnVTx has been broadcast without a child (SpendTx).

7. (OPTIONAL) **Emergency Deep Vault**:
   While the Vaults are sufficient against most attacks, if any of the stakeholder become compromised an EmergencyTx SHOULD be used. Doing so will result in the funds to move to the EDV, being only spendable with hard to access keys. The implementation SHOULD have a "panic button" broadcasting all the EmergencyTxs to empty the Vaults and UnVaulted outputs, resulting in a move of all the funds in custody to the EDV.


A new deposit is received
Each stakeholder is notified to sign UnVTx, CancelTx and EmergencyTx
Once all 4 have pre-signed, the Vault becomes spendable in the traders wallet
The 2 traders (or 1 trader + C) sign a SpendTx to move funds
The co-signing server verifies if the UnVTx is not yet mined.
The co-signing server verifies if a transaction from the same UnVTx has not yet been signed
If UnVTx mined: broadcasts CancelTx
If not, the co-signing server signs the SpendTx and Broadcasts with UnVTx
The stakeholders receive a notification that a SpendTx is broadcast
Once the UnVTx is mined, a delay of 6 blocks in required before SpendTx is mined
If there is a problem with the SpendTx, any stakeholder can broadcast a CancelTx
In case of grave compromise, EmergencyTx is broadcast
6 blocks later, the spending is complete.

Cosigning server

There is currently (Early 2020) no way to restrict the output of a child transaction in its parent locking condition. This means anybody capable to fulfill the spending conditions (typically: signatures) can generate competing child transactions with different outputs (or recipients). Only one of these will be mined.

This is a problem in our architecture as an attacker could sign a legit transaction, share it to the other stakeholders who will not broadcast the CancelTx nor EmergencyTx as everything seems right. When the OP_CSV expires, the attacker could broadcast a different transaction to the network, trying to double-spend the original SpendTx.

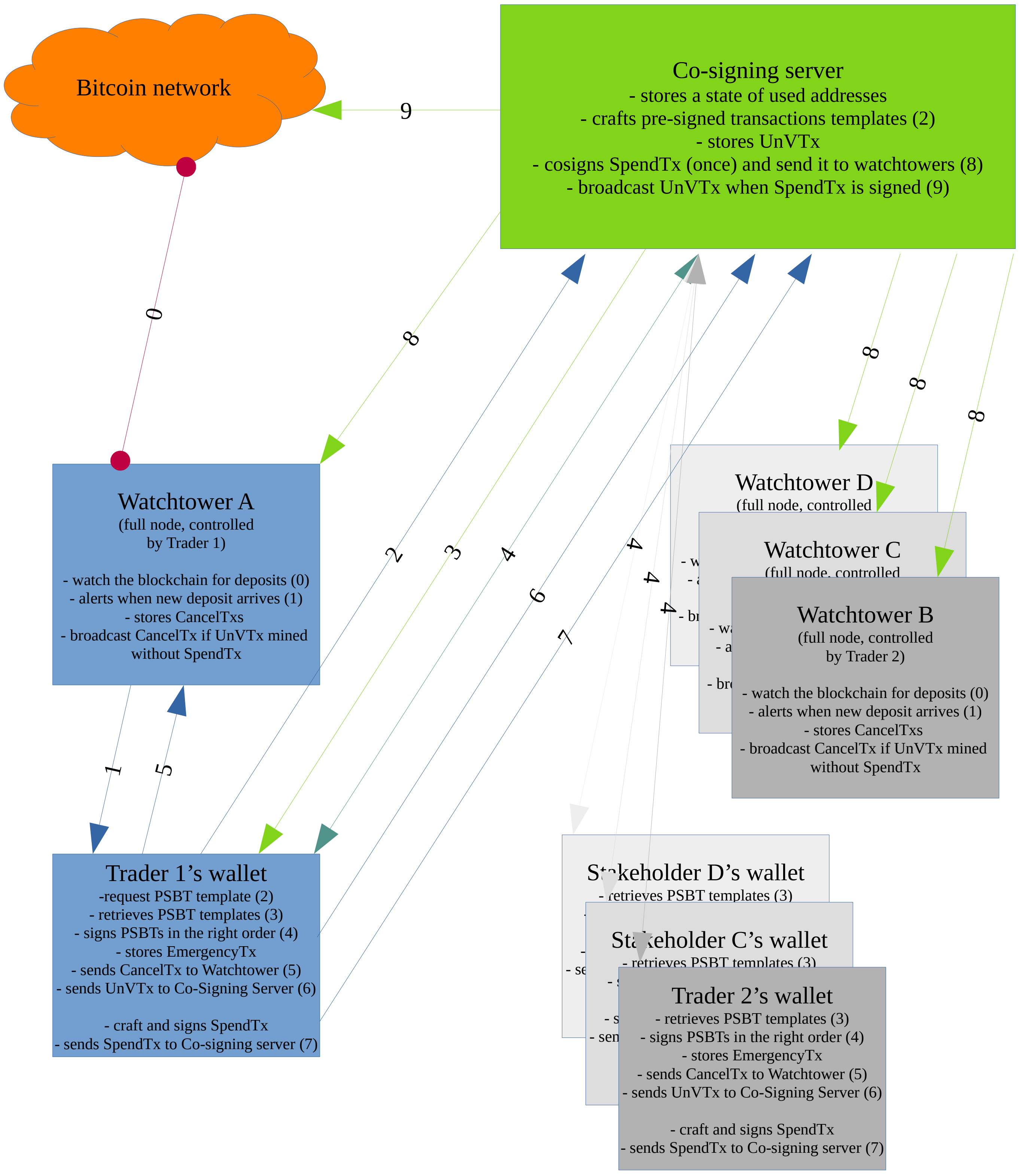
We add a co-signing server to enforce that only one SpendTx can be signed per UnVTx output. Therefore, the SpendTx shared with the stakeholders is the only SpendTx.

Replace By Fee (RBF) SHOULD also be disabled in SpendTx as an additional policy measure.

The cosigning server MUST have tamper-evident security measures, such as broadcasting an alert if an SSH connection or any change is detected.

A proposed improvement to this architecture is to have 1 cosigning server per stakeholder. This severely increases the costs of SpendTx but helps with the trustlessness of the system. Implementing such a model wouldn't be too difficult as we already need to share the SpendTx to the stakeholders at the time of UnVTx broadcast. Watchtowers will take this role and could also have an automated co-signing function to avoid the double-spend problem of the SpendTx, without centralizing this function to a single stakeholder.

Infrastructure


Bitcoin network
Co-signing server
- stores a state of used addresses
- crafts pre-signed transactions templates (2)
- stores UnVTx
- cosigns SpendTx (once) and send it to watchtowers (8)
- broadcast UnVTx when SpendTx is signed (9)
9
0
8
8
8
8
Watchtower A
(full node, controlled by Trader 1)
- watch the blockchain for deposits (0)
- alerts when new deposit arrives (1)
- stores CancelTxs
- broadcast CancelTx if UnVTx mined without SpendTx
Watchtower D
(full node, controlled
Watchtower C
(full node, controlled
Watchtower B
(full node, controlled by Trader 2)
- watch the blockchain for deposits (0)
- alerts when new deposit arrives (1)
- stores CancelTxs
- broadcast CancelTx if UnVTx mined without SpendTx
2
3
4
6
7
4
4
4
1
5
Trader 1's wallet
-request PSBT template (2)
- retrieves PSBT templates (3)
- signs PSBTs in the right order (4)
- stores EmergencyTx
- sends CancelTx to Watchtower (5)
- sends UnVTx to Co-Signing Server (6)
- craft and signs SpendTx
- sends SpendTx to Co-signing server (7)
Stakeholder D's wallet
- retrieves PSBT templates (3)
Stakeholder C's wallet
- retrieves PSBT templates (3)
Trader 2's wallet
- retrieves PSBT templates (3)
- signs PSBTs in the right order (4)
- stores EmergencyTx
- sends CancelTx to Watchtower (5)
- sends UnVTx to Co-Signing Server (6)
- craft and signs SpendTx
- sends SpendTx to Co-signing server (7)

Costs

The normal transaction flow cost, assuming a whole UTxO being spent, is currently around 1100 Weight Units, or 275 vbytes.

Transaction fees for the pre-signed transactions are an important subject to consider. The UnVTx needs to be relatively fast to mine, but not overpay too much as it is a necessary component of each transaction flow. RBF can be implemented and the wallets would automatically propose to sign the pre-signed transactions again if the network fees are higher than existing UnVTx fee-rate. Another approach would be to pre-sign multiple transactions with different fee-rate and let the wallet broadcast the appropriate one when needed.

The CancelTx and even more critically the EmergencyTx are subject to spam attacks. An attacker can fill the mempool (transactions waiting to be confirmed) with high-fee transactions to prevent the CancelTx and EmergencyTx to be mined in time (before expiration of the CSV, here 6 blocks).

Therefore it is necessary to have high fees on the CancelTx (we recommend 100 sat/vbytes) and unreasonably high fees on the EmergencyTx (1000 sat/vbyte). As CancelTx and EmergencyTx are rarely-if-ever supposed to be used, overpaying fees is acceptable.

Another consideration is around "which" mempool we use for the fee estimate. The SpendTx can be generated by the Traders. In our current proof of concept, the cosigning server generates the transactions and select the fee. This is still subject to change.

Security measures

This section explains the role of each step in the architecture regarding security.

- **4/4 vaults**: To avoid the spending of unexpected funds, each signatory need to sign a transaction to make any deposit spendable. An example of attack without this security would be a quick withdrawal from an exchange, or large deposit from a client, being spent before being acknowledged by all stakeholders. With the 4/4 multi-signature, the stakeholders know when there is a large sum in the Vaults.
  Drawback: funds are not available if one of the signer refuses to or cannot sign. Making the scheme a 4/5 would help but also change the threat model.
- **Unvaulting transactions**: OP_CSV counts time since the output being spent was mined, not since the spending transaction was broadcast. A "hack" around this is to make sure both the parent and child transactions are broadcast together, enforcing the full delay

before the child is mined. There is no on-chain alternative to give a revocation delay from time of broadcast (as of Q1 2020)

- **Co-signing Server**: The SpendTx transaction still has an exploitable vulnerability due to how OP_CSV works. The two Traders could double-spend themselves at the block height the delay expires. This attack cannot be performed easily. To mitigate this risk, an automated third signer is added, which signs an input only once.
  This is similar to signing-then-deleting-key which cannot be proven in a multi-party environment.
- **Emergency Vault**: A multi-signature address with separate keys. The keys are not used anywhere else in this architecture and should be stored in secure and hard to access location. This probably will not be used but helps against physical threat or compromise of the infrastructure or it's users.
  Drawback: It should be really annoying to spend from. This is a feature but can be used maliciously to disrupt business operations by any participant or somebody getting access to a pre-signed Emergency Transaction.
- **Watchtower**: Because spending from a Vault uses a pre-signed transaction and needs only a 6 blocks delay to be mined, it is reasonable to assume there will be time when nobody sees and verify the Spending Transaction before it is confirmed. This could be used to steal funds when the stakeholders are known to be away or busy. Watchtowers can be deployed to automate some of the security, for example verifying the recipient is an exchange, amounts are within a limit per unit of time, or broadcast is during office hours.
- **Pre-signed transaction fees**: A compromised SpendTx could try to bypass the CancelTx and EmergencyTx by spamming the mempool and preventing revocation. With an EmergencyTx paying 1000 sat/vbytes of transaction fee, an attack would cost more than 240 bitcoin to pull off (6 full blocks of more than 1000 sat/vbytes). This fee needs to be aligned with the funds held within vaults, and reevaluated regularly.

Key Management

This architecture doesn't involve a third party. Each stakeholder is responsible for their keys, this section is an incomplete set of recommendations to help with key management.

Keys should be kept offline, either on a computer with no internet access, or on a dedicated hardware wallet. This also implies that keys MUST NOT be stored, even temporarily, on a device with internet access.

New keys SHOULD be generated during a ceremony, to make sure all participants use proper key generation methods. Chainsmiths recommends a dice + external entropy source generation, with a log of the intermediary steps (dice rolls and entropy used). The log will be stored with the key backup, and should never be revealed unless for forensics in case of loss of funds.

Key backups MUST be sealed in a tamper evident manner. Verification of the seals MUST be performed regularly. If a key is suspected of being compromised, a key rotation MUST be performed immediately.

The Vault keys MUST have a recovery plan in case of disappearance of the stakeholder. This plan is left to the discretion of each stakeholders for their own keys. It is critical that the stakeholders do not centralize the backups (for example, they MUST NOT all use the same notary, safe deposit box location, etc).

The EDV keys MUST NOT have a recovery plan in case of disappearance. Each stakeholder MUST ensure they are the only ones to know where the keys are stored, and it must be very hard or slow to access. There also MUST be at least one backup key, in a different location. These keys should be treated as the utmost secret, we recommend the owners to not leave traces of where it could be located (don't stay at a hotel, don't use a personal bank card, don't bring your phone, don't log in to any online account, etc).

Discussions/reviewer comments

a) The architecture currently gives access to all pre-signed funds to the traders. A successful attack would empty all of the Vaults. To mitigate this, the Unvaulting transactions should only be provided to the traders for amounts they need, not more. There are multiple ways to add this logic in the wallet infrastructure, either on the stakeholders wallets directly, or as rules for the co-signing server.

b) While the risk of theft is reduced, there exist a risk of ransom from one of the stakeholders refusing to sign new deposit. We assume in this model that the stakeholders are identified and the legal system is enough to deter against such an attack. Automated checks that incoming deposits are signed regularly should be added to detect such an anomaly before the locked amounts are too high. (i.e. if a deposit hasn't been signed for more than a day, an alert should be issued to avoid generating new deposits)

c) A "heart-beat" check of the system should be performed when no deposit are received for multiple days. This can take the shape of small transactions or a message to sign, verifying the system is still functioning properly and stakeholders are still signing.

d) The 6 blocks delay represents about 1 hour on average, but can be much faster/slower due to the variance of mining. This delay is still relatively short, leaving little time to broadcast a Cancelling Transaction. Longer delays are safer but impact the speed of transactions. This should be made as long as possible to ensure security.

e) Key management for this architecture is complex. It should be seen as a weakness of this proposal, especially for users without prior experience with cryptographic key management or other high security storage.

f) The architecture has a lot of moving parts, clients should consider if they want such a complex system with potential down-time or if they can reduce the security and go with a simpler set-up (typical multisig with a cosigner).

Next steps

This architecture is currently going through a technical proof of concept and additional peer-reviewing.

Given the lack of compatibility with major Bitcoin software and the need for long term support, a dedicated entity (non-profit) will be set up to support contributors of the libraries and relevant updates to Bitcoin. This entity is still being conceptualized but will be oriented towards institutional Vault protocols.

This architecture is built to be functional with contemporary Bitcoin. It will evolve with the Bitcoin protocol, specifically when Schnorr, OP_CTV, sighash_anyprev or similar changes will be implemented to the core protocol.