# Bitcoin Developer Community Workshop

## Meeting Minutes

November 4, 2021 14:00-16:00 CET via Zoom

This document captures the main discussions of the meeting, such as answers to technical questions by the community, discussions about use cases, testing, and further requirements to consider for the implementation of the feature.

## Technology Recap

Threshold ECDSA (t-ECDSA) key management

- Final decisions for t-ECDSA master key management are still to be made.
- What is clear now is that all canisters on a single subnet will be deriving their keys from a master key on this subnet.
- We are looking at tradeoffs between security and performance.
    - When deploying the same master key on multiple subnets, the security for the key will be the minimum of the security provided by all the subnets the key is deployed on. This implies that we need to use only sufficiently secure subnets to host a given master key.
- Current strategy. Having a master key deployed on multiple subnets to ensure we have a backup.

Can one sign and send any arbitrary Bitcoin transaction e.g., without using t-ECDSA for signing?

- Yes, we are not constraining the transactions one can submit. E.g., you can submit a transaction signed by someone else, e.g., users, and not necessarily through a canister with a t-ECDSA key.

Is t-ECDSA decoupled from Bitcoin functionality?

- Yes, it's completely decoupled: It is built as a separate feature that provides an API that Bitcoin smart contract canisters use to obtain public keys and sign transactions. Using the t-ECDSA API is the whole integration between the Bitcoin and t-ECDSA features. Any canister can also use the t-ECDSA signing functionality for any purpose other than Bitcoin.

Which of the Bitcoin API calls are (a)synchronous?

- According to the current plan (not fully decided yet) all API calls are asynchronous, but this is not strictly required. Some APIs could be offered as synchronous calls.
- Note that asynchronous calls are very fast in this setting because the Bitcoin functionality is deployed on the same subnet as the calling canister.

Is Bitcoin finality accounted for in the get_UTXO_set API?

- Yes, you get a confirmation count as metadata for a UTXO in the UTXO set for a Bitcoin address returned by our API.
- For the library we plan to build, Bitcoin finality will probably be abstracted even further so you don't need to handle confirmation counts for individual UTXOs yourself, but only specify the minimum confirmation count you want to have for UTXOS.

What is the status of the feature allowing canisters making http calls?

- This is also high on our roadmap and will eventually be built, but currently our focus is on Bitcoin integration.

Will canisters be able to hold bitcoin before ICP?

- No, they will be able to hold ICP before they will be able to hold bitcoin.
- ICP on canisters is currently being worked on aggressively. We will start to see a v0 of DeFi on IC in Q1 2022.

How does the relaying between the Internet Computer and the Bitcoin network work? Is there a set of Bitcoin nodes that we connect to?

- This is realized by our own IC nodes in an as decentralized way as possible: Each node of a Bitcoin-enabled IC subnet connects to a set of random nodes of the whole Bitcoin network. We don't constrain ourselves to a subset of the Bitcoin network, but randomly discover the nodes to connect to from the full Bitcoin network.
- This approach allows for quickly spreading outgoing transactions to many nodes of the Bitcoin network in a highly decentralized way as well.

## Use Cases

A forum member asks the question on whether we would need wrapped Bitcoin for an AMM project. Without wrapped Bitcoin, one would need to wait for the confirmation time on the Bitcoin network, which takes too long. He postulates that almost all use cases will still need some form of wrapped Bitcoin due to their short finality time. Is the Foundation covering this wrapped Bitcoin functionality or is the community expected to do this?

- People agree that we will need some form of wrapped Bitcoin on the IC.
- We (the Foundation) have it on our agenda for Phase 2 of the Bitcoin integration, but would be happy as well if the community could help here.
- With the basic integration (Phase 1) it should be possible for everyone to build a wrapped Bitcoin canister.
- Technical options for realizing wrapped Bitcoin:
    - Token: When a user transfers bitcoin to the canister, it mints wrapped bitcoin tokens to the user.
    - Ledger: Users can transfer bitcoin to a ledger canister. When someone makes a transfer, the canister keeps track of the transfer in a ledger. We are building an open source ledger canister that can be used as a basis for implementing the ledger option. With this, it should not be too much work to implement this option.
- We conclude that if there is a huge demand for wrapped Bitcoin on the IC, it is probably better if the Foundation implements it for reasons such as trust. Otherwise, additional trust is required in the ones who build the wrapped Bitcoin functionality.
- The community already creates wrapped versions of ICP to get rid of the account id paradigm so they can take a principal-id-centric approach which they want for some of their use cases. If the Foundation will implement the wrapped Bitcoin implementation in the account id scheme, there will probably be other wrapped versions that emerge anyways or people may create their own versions.

Drivers behind this may be that one wants to use principal ids, i.e., for the same reasons that ICP is wrapped. This is, e.g., required for the mentioned AMM project.

How can the account balance for the user be accessed when natively holding bitcoin in a canister? Which API can be used? Would the resulting balance be the same balance you see in any block explorer?

- The get_balance method in the library or get_UTXO_set system API can be used for this. They can be invoked for any Bitcoin address to obtain the respective balance or UTXO set.
- The balance one obtains via this matches exactly with what you see on a Bitcoin blockchain explorer. (we use Bitcoin blocks to extract the transactions and UTXOs based on which the balance is computed, so it's the exact same view as in a blockchain explorer).

How can you link a Bitcoin address to a principal id?

- You can send bitcoin to a Bitcoin smart contract and based on the received funds mint tokens to the principal id of the sender of the bitcoin.
- This functionality can be easily implemented in a canister. The system is powerful enough to allow you to implement something like that.

How can a canister verify that bitcoin comes from a particular user?

- Option 1: Give every user their own Bitcoin address created using BIP-32 key derivation. If a transfer is received on a particular address, one can assume it is from the user principal with whom this Bitcoin address has been associated.
- Option 2: Alternatively, if we want to only use one address: We can use Bitcoin opcodes to encode the relevant information in the Bitcoin transaction.

What is the timeline?

- The original plan was to release by the end of the year 2021.
- The dependency on t-ECDSA will lead to a delay for the ECDSA-integrated release as t-ECDSA will slip into Q1 of next year. However, it is possible, see below, to make available parts of the functionality without t-ECDSA to already facilitate smart contract development until t-ECDSA is ready.

Discussion on testing when developing a canister that uses Bitcoin functionality: How can smart contract developers do testing for their canister code in their local environment? E.g., ensuring that the correct Bitcoin transactions are going out?

- The Foundation has a similar challenge during development of the feature and they are thinking about how the ideas used can be transferred to canister development. They will try to make some of the tools built for their own testing available so that developers can test their code before deployment. This is an area still to be further explored.
- We could allow interaction with the Bitcoin testnet to help people test run their canisters without making transactions on the Bitcoin mainnet. Providing a toggle for selecting between Bitcoin mainnet and testnet.
    - However: It is tricky to get Bitcoin testnet tokens, maybe running one's one Bitcoin test network offline is a better approach.
- Integration offline would be advantageous: E.g., using a mock Bitcoin network running locally. Likely this is easier to implement and to use for developers than the Bitcoin testnet. Enables "development on the beach" without an internet connection.

Developer preview of the Bitcoin feature: We could release the Bitcoin feature without threshold ECDSA being finalized for accelerating their smart contract development early on.

- People are excited about the idea of putting out something for developers to use before we have t-ECDSA finished so that people can already start implementing their smart contract canisters against the Bitcoin API. This would go hand in hand with other tools we make available for local testing. The community agrees that this is something that we should definitely consider doing to facilitate development by the community early on.
- N.B.: The IC does not have a testnet because cycles are so cheap.

A participant from the community mentioned that there are projects out there that do not want to use bridges for Bitcoin integration and for which our approach of direct integration would be a great match. We found that it would be a great idea to connect to such projects.

A use case mentioned was cross-border payments, e.g. for merchants, for geographies where local currencies are not very stable. This kind of integration makes sense for such a scenario when one would want to hold Bitcoin rather than local currency.

- One of the key aspects would be to have a way for a canister to be controlled by a group of people rather than a single address: Keeping a canister as a separate entity with multiple users owning it jointly. Bitcoin on the canister belongs to those different users and would release programmatically based on defined rules.
- This is something that can be done pretty easily on the IC.

Is the Bitcoin team the same team that will be doing the Ethereum integration?

- Likely there is a large overlap with the team, but the team is not defined yet.
- We are already thinking about Ethereum now while engineering efforts for Bitcoin are ongoing.

EVM / Solidity support on IC Community members expressed the strong feeling that not only for Ethereum-related projects, but even for Bitcoin projects EVM compatibility is important.

- Projects often develop smart contracts in Solidity and it is infeasible to reimplement them in a different language and get liquidity within a reasonable timeframe. The capability of being compatible with EVM / Solidity is important to receive liquidity through such projects.
- Releasing such EVM / Solidity compatibility on the Internet Computer alongside the native integrations is considered to be crucial. The IC would immediately become the best layer 2 solution for Ethereum and Bitcoin given most or the use cases are written in Solidity.
- The community is willing to help with the implementation. What is blocking us from the community helping with this feature?
    - Knowing how native Ethereum integration is going to work would be extremely helpful to make sure that this EVM environment works really seamlessly with that.
    - Canister heap limit of 4 GB might be an issue, depending on how big EVM implementation is combined with the state of whatever contract is running.
        - More stable memory is not enough, we need 64-bit heap addressing.
- Some approaches taken by other blockchains for EVM compatibility:
    - NEAR: https://aurora.dev/about
    - Solana: https://github.com/hyperledger-labs/solang
- Possible approaches for the IC

    - Use an EVM compatible runtime on the IC: This causes problems with synchronized execution as the Wasm runtime is message based, but contract calls are synchronous.
    - Compile Solidity contracts to WebAssembly and deploy them on the IC: This requires an according concurrency model to be implemented in order to achieve the synchronous semantics of Solidity on the asynchronous IC.
- Next steps
    - We need to organize a community workshop on this topic.
    - We will coordinate between people on how to keep EVM integration discussion going, e.g., on the Forum.

## BIP-32 Key Derivation

BIP-32 key derivation

- Canister root key is derived using index 0 of BIP-32 key derivation.
- Canisters can use other indexes to derive an arbitrary number of public keys.
- All the derivation we use is unhardened key derivation.

Motivation for using BIP-32 key derivation and not using independent key pairs?

- Maintaining a threshold private key is expensive. Every independent private key is secret shared among the replicas of the subnetwork and re-shared every couple of minutes for higher security. It has been an early decision to have as few independent keys as possible.
- Key derivation solves the problem of having few keys.
- The requirement to be able to perform disaster recovery on the key also drives cost.
- As we need key derivation anyway for those reasons, we get BIP-32 cheaply on top of it.

Linkability of public key addresses

- This has some security implications regarding anonymity: Targeted attack on an address, we know the subnet and node operators of the subnet.
- To counter attacks of this type, we run t-ECDSA on our most robust, largest subnets.

Could we shard assets of a subnet if there's too much bitcoin on it?

- Would not matter a lot: If the Bitcoin subnetwork has the same number of nodes as the NNS network, this is as good as it gets. If you control the NNS subnet, you control the other subnets as well.
- We can add more nodes from more independent node providers for better security.

Others use BIP-32 as well for deriving keys, no issue is seen here.

- From a wallet perspective, people expect BIP-32 key derivation, so it's the right thing we are doing.

Multisignature wallets on IC, some rough ideas that came up:

- Threshold signatures on IC would be a cool feature.
- Don't think we need Taproot, but can abstract it away.
- Could do multisignatures for multichains with EVM integration.
- Requires more thought

## Next steps

- Organize follow up.

Continue discussions on EVM in forum, either in Bitcoin topic or new topic.