A Decentralized Mechanism for Know-Your-Transaction Compliance

Thomas Locher, DFINITY Foundation
thomas.locher@dfinity.org

Extended Abstract

Know-your-transaction (KYT) compliance mandates the use of control mechanisms that identify suspicious customer actions in an effort to combat and prevent financial crimes such as money laundering. This type of regulation also affects cryptocurrency exchanges, which, in addition to know-your-customer (KYC) verification, also utilize services to perform KYT checks. Since Bitcoin is by far the largest cryptocurrency with respect to market capitalization, it is the focal point of KYT, and multiple companies such as Chainalysis, Elliptic, and CipherTrace provide services to classify bitcoins based on their transaction history. Concretely, these KYT service providers offer subscription-based APIs that, given a Bitcoin address or a specific transaction output, return their classification measuring the "exposure" to fraudulent activity. If there is any exposure, the Bitcoin address or transaction output is considered "tainted". If a reputable exchange and financial institution receives tainted bitcoins through a customer transfer, the incident is reported and the bitcoins in question are likely to be confiscated. While the crypto community may see Bitcoin as a fully fungible token, this is not the case in practice.

Bitcoin users who trade exclusively on centralized exchanges are typically oblivious to KYT checks running in the background, and they are unlikely to ever receive tainted bitcoins because the exchanges monitor their funds vigilantly. However, decentralized marketplaces and other decentralized finance (DeFi) products bring about new risks for users. DeFi applications usually do not perform any kind of KYT checks for multiple reasons.

First of all, decentralized applications (dapps) are normally not classified as virtual asset service providers and therefore not obliged to adhere to regulatory requirements. Moreover, dapps executed on self-contained blockchain platforms simply lack the capability to perform KYT checks. The risk for a user is that bitcoins obtained from such dapps may be considered tainted by exchanges, causing trouble for the user after transferring the tainted funds to an exchange.

This article presents the first decentralized dapp that is KYT-compliant, significantly reducing the aforementioned risk for all users. The dapp is the so-called chain-key Bitcoin (ckBTC) minter, a smart contract that handles the conversion between bitcoins and ckBTC, a token on the Internet Computer that is backed 1:1 by bitcoins such that 1 ckBTC can always be redeemed for 1 bitcoin and vice versa. Since ckBTC tokens can be transferred within 1-3 seconds and each transfer merely costs the equivalent of 10 satoshi, ckBTC provides a fast, inexpensive, and decentralized way to trade bitcoins, and is therefore appealing to frequent traders.

The Internet Computer has a unique feature in the crypto space that makes KYT checks possible for dapps: Smart contracts can issue HTTPS calls to regular web servers and process the responses. This feature is used in a specific smart contract that interacts with a KYT service provider to classify Bitcoin addresses and transaction outputs. The ckBTC minter calls this smart contract to perform KYT checks for both deposits and withdrawals. More precisely, it requests a KYT check for the corresponding unspent transaction output from the KYT smart contract before accepting a deposit, and asks the KYT smart contract to verify that the Bitcoin address where funds are meant to be transferred is "clean" for withdrawals, ensuring that ingress and egress flows are protected.

In this article, we describe this decentralized KYT mechanism in depth. We discuss advantages of this approach as well as challenges that open up interesting avenues for future work.

1 Web3 Services & KYT Compliance

Web3 is often touted as the "next generation" of the World Wide Web, augmenting the current Web2 with concepts such as decentralization and token-based economics ("tokenomics") that enable creators to retain ownership and control over their content, thereby reducing the power of centralized tech companies. Due to Web3's potential to democratize the digital space, interest in Web3 services has been on the rise for several years. In fact, Web3 equivalents for a plethora of Web2 platforms have emerged in numerous areas including gaming, social media, and, most prominently, financial services. Tokenomics play a pivotal role in Web3 as many services manage or interact with virtual assets, typically in the form of tokens. Classic examples are decentralized exchanges (DEXs), rivaling their centralized counterparts and attracting users with low fees, and decentralized market places for digital goods. DEXs started out as simple automated market makers, using liquidity pools of crypto tokens for trading, but full order-book DEXs are emerging on the market, nearing feature parity with CEXs.

Apart from the lack of centralized control (and possibly lower fees), a crucial difference between CEXs and DEXs is that DEXs generally lack regulatory oversight, which entails a substantially higher risk of financial crimes including money laundering. In the Web2 world, companies such as Chainalysis [1], Elliptic [2], and CipherTrace [3] that track cryptocurrency flows and link them to criminal activities have established themselves as business partners of CEXs and financial institutions in general. These companies provide APIs to their (paying) customers to check if any source or destination address has had direct or indirect exposure to any recorded illicit activity, enabling the customers to accept funds or approve outflows in accordance with legal standards and regulations. While presumably all major CEXs implement a KYT strategy, the opposite is true for DEXs and other services in the Web3 space.

The Financial Action Task Force (FATF) defines a virtual asset service provider (VASP) as a business that engages in the management, safekeeping, exchange, and transfer of virtual assets [4]. This definition includes cryptocurrency services such as exchanges, wallet custodians, and many others. It is important to note that, if there is a transfer of value, even a decentralized application (dapp)

and/or its owner/operator(s) may be considered VASPs with obligations to carry out KYC and KYT checks. However, if a dapp does not have an operator, it is unclear what party should be considered the corresponding VASP, if any. What is more, most blockchains are self-contained systems that cannot initiate any interaction with external entities. Consequently, it is usually infeasible or undesirable for a dapp to perform such checks. A cumbersome approach would be to keep all transactions pending until they are fetched by an external oracle service that passes them on to a KYT service provider for verification before returning the verification results to the dapp. In this approach, the dapp becomes entirely dependent not only on the KYT service provider but also on the oracle service, increasing the complexity and costs of the dapp while severely hampering both performance and decentralization. On the other hand, the absence of KYT checks puts the owners or operators of the dapp in a legal gray zone—and the users may not be fully aware of the risk that they are exposed to when interacting with the dapp.

2 The Internet Computer & HTTPS Outcalls

As mentioned in the previous section, a crucial shortcoming of most blockchain platforms is the lack of any means to access information external to the blockchain. This shortcoming is due to the fact that blockchains are deterministic replicated state machines, where the same changes must be applied in the same order to ensure that any state transition leads to the same new state. If the machines running the blockchain obtained different results when querying the same external data source, consensus would have to be reached first as to which result to use for the state update.

The only blockchain platform that offers such a feature is the Internet Computer [5], a general-purpose blockchain-based platform that has several unique capabilities [6]. The main feature of interest for this article is the capability to make HTTPS outcalls [7] from inside smart contracts. The function `http_request` that any smart contract can call to access data from the web is defined as follows:

```
type http_response = record {
  status: nat;
  headers: vec http_header;
  body: blob;
};

http_request : (record {
    url : text;
    max_response_bytes: opt nat64;
    method : variant { get; head; post };
    headers: opt http_header;
    body : opt blob;
    transform : opt record {
      function : func (record {
        response : http_response; context : blob})
        -> (http_response) query;
      context : blob
    };
  }) -> (http_response);
```

The request must contain the targeted URL in the `url` field. Additionally, the access method must be specified, which is either, `get`, `head`, or `post`. In addition to the status, the response contains the result in the `body` field in the form of a byte array (`blob`). As the definition shows, there are more fields in the request and response. Since this is not the core topic of this article, we dispense with a more detailed description and refer the interested reader to online documentation [8]. It is worth noting, however, how the Internet Computer ensures deterministic state transitions given that the machines may receive different responses. In short, the protocol simply verifies that at least two thirds of all responses are identical. If this is the case, the response is accepted and returned to the calling smart contract. Otherwise, an error is returned. Since responses from web servers can contain various metadata such as unique request identifiers, timestamps, and other data that is likely to change with every request, a transform

function can be specified that extracts the relevant piece of information from the response before comparing for equality, thereby greatly increasing the chances of getting an agreed-upon response.

Obviously, HTTPS outcalls cannot be used to access rapidly changing data because the machines would rarely be able to reach consensus. Fortunately, KYT data does not change quickly, which makes it possible to utilize this feature to implement a decentralized KYT mechanism.

3 Chain-key Bitcoin & KYT Verification

Bitcoin is still the commanding force in the cryptocurrency space, yet it is seldom used for decentralized finance (DeFi) applications due to its lack of programmability, high transaction fees, and low transaction speed. While the Lightning network [9] is intended to reduce fees and improve transaction throughput, several projects such as Stacks [10], Rootstock [11], and chain-key Bitcoin (ckBTC) [12] that enable more general-purpose smart contracts to be built on Bitcoin have recently grown in popularity.

In this article, we focus on ckBTC, a token on the Internet Computer that is backed 1:1 by bitcoins (BTC) and that addresses the aforementioned drawbacks: ckBTC can be used inside smart contracts implementing arbitrary business logic, every transaction costs the equivalent of 10 satoshi, i.e., 0.0000001 BTC, and transactions settle with finality typically after 1-3 seconds.

Naturally, a 1:1 peg can only be maintained by guaranteeing that 1 ckBTC can always be redeemed for 1 BTC and vice versa (minus fees). To this end, the ckBTC functionality is split into two main smart contracts:

1. The *ckBTC minter* is responsible to mint ckBTC when a user deposits BTC and burn ckBTC when a user retrieves BTC.

2. The *ckBTC ledger* executes all ckBTC transactions. Moreover, it mints and burns ckBTC as instructed by the ckBTC minter.

The ckBTC minter makes use of another unique capability of the Internet Computer, its bridgeless integration with the Bitcoin network [13] that enables a

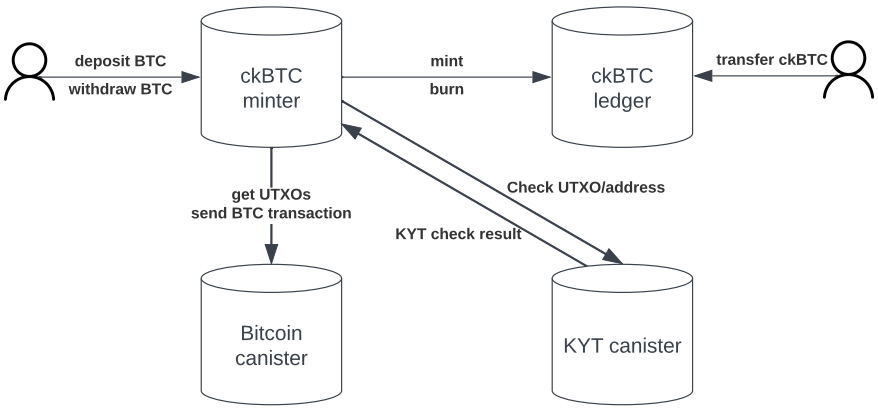

Figure 1: An overview of the interactions between users and the smart contracts that together provide the ckBTC functionality.

smart contract to hold, receive, and send real bitcoin. Since no bridge is required to connect the two networks, it is a more secure alternative to using wrapped tokens such as WBTC on Ethereum [14]. The ckBTC minter effectively takes all received bitcoins under custody and instructs the ckBTC ledger to mint and burn ckBTC tokens whenever bitcoins have been deposited and retrieved, respectively. A high-level overview of the architecture is shown in Figure 1.

In the lower left corner, the *Bitcoin canister* is the system-level functionality that provides read and write access to the Bitcoin blockchain through endpoints that return the unspent transaction outputs (UTXOs) and balances of Bitcoin addresses (read access) and accept Bitcoin transactions from smart contracts, which are then advertised in the Bitcoin network for inclusion in a block (write access). Note that a smart contract executed on the Internet Computer is referred to as a *canister*, which is a bundle of the smart contract logic and the data that holds the smart contract's state.

It is worth noting that another technical feature is required: A smart contract must be able to have at least one Bitcoin address and it must be able to securely sign transactions to spend funds associated with its addresses. This functionality is enabled through threshold ECDSA [15], another feature available on the Inter-

net Computer: Smart contracts can obtain one or more ECDSA public keys and request signatures for a given piece of data and one of these keys. Since several types of Bitcoin addresses simply encode an ECDSA public key and spending bitcoins requires signatures for the encoded public keys, threshold ECDSA indeed provides the required functionality.

We are now in the position to return to the main topic of KYT compliance. This part is covered by the *KYT canister* shown in the lower right corner, a smart contract used by the ckBTC minter as follows. When the ckBTC minter receives a new UTXO, it sends a request to the KYT canister to perform a KYT check on the UTXO. Similarly, when there is a BTC retrieval request to a certain destination address, the ckBTC minter asks the KYT canister for a KYT check on the Bitcoin address. Since KYT checks are performed when bitcoins are received and transferred out, both ingress and egress flows are protected.

The KYT canister cannot answer the requests itself. Instead, it uses the HTTPS outcalls feature to make requests for KYT checks to a KYT service provider. The response of the KYT service provider is then simply returned to the ckBTC minter. The response may contain a list of alerts associated with the given UTXO or address with each alert having a certain severity level. The ckBTC minter treats alerts conservatively in that it marks UTXOs and addresses as tainted if there is any alert, regardless of severity. If a UTXO is tainted, it is quarantined and no ckBTC are minted. This UTXO is then "stuck" in the ckBTC minter since there are currently no clear guidelines as to how tainted bitcoins are to be treated. If a Bitcoin address is tainted, on the other hand, the request to send bitcoins to this address is simply declined and the user who made the request retains his or her funds in ckBTC.

An important technicality that must be considered is the fact that KYT providers offer paid subscription services. Since smart contracts cannot purchase subscriptions using fiat currencies, there is a need to introduce the role of a *maintainer*, which is an entity that has acquired a subscription and registered the corresponding API key (required to gain access to the KYT service) with the KYT canister. When registering an API key, the KYT canister uses the API key to make a KYT request against the KYT provider to verify that the key is valid before accepting the maintainer. For each request, the KYT canister picks a reg-

istered maintainer and uses the maintainer's API key to make the request to the KYT provider. For each successful request—irrespective of the outcome of the request—, the maintainer gets awarded a KYT fee of 0.00002 ckBTC. The ckBTC minter remunerates the maintainers with a daily lump-sum payment into their accounts on the ckBTC ledger. These payments are meant to offset the cost of purchasing the subscription. The user depositing or withdrawing bitcoins must pay the KYT fee, in addition to the Bitcoin network fee and a small fee that goes to the ckBTC minter as compensation for the cost incurred to send the Bitcoin transaction.

When looking at the figure, the KYT mechanism appears deceptively simple but multiple complex features need to interact seamlessly and several technical details need to be considered to make the whole process work.

4 Open Challenges

Overall, the mechanism described in the preceding section makes it possible to use ckBTC as a fast and cost effective "twin" of Bitcoin with confidence that every ckBTC token is worth one bitcoin because a) the underlying asset is held by the ckBTC minter itself and b) every accepted UTXO underwent a KYC check. It is therefore improbable that a user would ever run into issues when sending retrieved bitcoins to a centralized exchange or a financial institution.

However, there are some open challenges. First of all, only one KYT provider is used currently. The damage this KYT provider could cause is limited in that it does not have control over any funds. In the worst case, the KYT provider would return erroneous information, causing the ckBTC minter to accept tainted UTXOs or send bitcoins to tainted addresses, or reject UTXOs and addresses even though the UTXOs and addresses are clean. In the latter case, the KYT provider would disrupt the minting and burning process but regular ckBTC transfers would not be affected. The KYT provider constitutes a single point of failure in an otherwise decentralized architecture. Fortunately, the risk of this scenario is low because a reputable KYT provider can be expected to reliably provide correct data. More importantly, in the unlikely event that the KYT provider ceases to provide its

service, the decentralized governance system of the Internet Computer can be used to upgrade the KYT canister in order to switch to another KYT provider, i.e., this dependency cannot make the ckBTC minter inoperable. Clearly, the KYT process can be decentralized in the future by having the KYT canister interact with various KYT providers, at the expense of (linearly) increased fees.

Another challenge pertains to maintainers: The process of registering maintainers and API keys is already fully decentralized. The challenge is to find maintainers willing to make the initial investment in fiat currency to get a subscription and to get reimbursed through daily ckBTC payments. Since the KYT fee is a fixed ckBTC amount, a maintainer can only turn the investment to profit if there is a substantial number of deposits and withdrawals coupled with a sufficiently high Bitcoin price. Therefore, it is expected that entities building their businesses around ckBTC are primarily interested in becoming maintainers. As there is no centralization concern if there is merely a single maintainer, it is therefore only required for the community to ensure that such a maintainer is always available.

Lastly, the management of API keys is another challenge because the machines running the KYT minter store the API keys. In theory, that puts the operators into a similar position as the KYT providers in that they could attempt to read out the API keys and try to deplete their quotas, which would make the mint and burn operations fail. Again, the risk of such an attack is deemed low because it is quite an effort to read out this information for little gain as the maintainers can use a single function call to replace their API keys. Furthermore, due to the scalable nature of the Internet Computer, only a small set of machines scattered around the world host the KYT canister. Consequently, it is not as easy to get access to such internal state information as it would be on other blockchain platforms. Nevertheless, even if the risk is low, it would be beneficial to have better protection of API keys in place. Since the API keys are used as part of the URL in HTTPS outcalls, it appears to be infeasible to hide the keys from the smart contract itself. Currently, the best available solution is to protect the entire smart contract by running it in an encrypted virtual machine. Since more and more machines are equipped with support for AMD's secure encrypted virtualization-secure nested paging (SEV-SNP) technology [16], it will be possible to add this layer of security in the near future.

5 Conclusion

To the best of our knowledge, the presented decentralized mechanism for KYT compliance is the first of its kind. As more and more funds are transferred in and out of dapps, it will become increasingly important to provide the means to protect users from financial crimes and having KYT checks baked into smart contrast is a step in that direction. As pointed out in this article, there are still challenges to overcome but we believe that we have only scratched the surface of what is possible. It is not unthinkable that full KYT services will eventually run on-chain, competing for users and providing KYT data to smart contracts directly in a more affordable, direct way without any sacrifices in terms of security or decentralization. Naturally, services for other types of compliance checks can emerge as well, making it possible for smart contracts to choose which checks are necessary for their use cases. Due to ckBTC and its KYT mechanism, decentralized compliance verification has just entered its infancy.

References

[1] Chainalysis. `https://www.chainalysis.com`.

[2] Elliptic. `https://www.elliptic.co`.

[3] CipherTrace. `https://ciphertrace.com`.

[4] Updated Guidance for a Risk-Based Approach to Virtual Assets and Virtual Asset Service Providers. `https://www.fatf-gafi.org/en/publications/Fatfrecommendations/Guidance-rba-virtual-assets-2021.html`.

[5] Internet Computer. `https://internetcomputer.org`.

[6] Internet Computer capabilities. `https://internetcomputer.org/capabilities`.

[7] Internet Computer HTTPS Outcalls. `https://internetcomputer.org/https-outcalls`.

[8] HTTPS Outcalls: Technical Wiki Page. https://wiki.internetcomputer.org/wiki/HTTPS_outcalls.

[9] Lightning. https://lightning.network.

[10] Stacks. https://www.stacks.co.

[11] Rootstock. https://rootstock.io.

[12] Chain-key Bitcoin. https://internetcomputer.org/ckbtc.

[13] Internet Computer Bitcoin Integration. https://internetcomputer.org/bitcoin-integration.

[14] Wrapped Bitcoin. https://wbtc.network.

[15] Threshold ECDSA: Chain-key Signatures. https://internetcomputer.org/docs/current/developer-docs/integrations/t-ecdsa.

[16] AMD SEV-SNP. https://www.amd.com/en/processors/amd-secure-encrypted-virtualization.