Excel@FIT 2023
http://excel.fit.vutbr.cz

Nummularius: Cryptocurrency Exchange with Trusted Computing

Bc. Tomáš Sasák*

Abstract
Most popular centralized cryptocurrency exchanges work in a way that requires the customer to fully rely on the exchange operators. Users should trust the operators that they will not interfere with the funds of the customers.
We propose a design of exchange with focus on security, using trusted computing, especially the Intel SGX. Using SGX enclaves, we can achieve transparency between the customer and exchange. Combined with the smart contract on public blockchain we achieve non-equivocation of the exchange.
The current proof-of-concept implementation is capable of handling around 35 customer deposits per second. In terms of bidding, the enclave is capable of handling around 23 bids per second for a single coin pair. The bottleneck occurs with the usage of Merkle-Patricia tree (trie) and other cryptographically oriented structures, since trie and other parts need exclusive access while being edited.
This work can be used as a stepping stone to use trusted computing hardware within centralized exchanges, blockchains, or other security-oriented systems.

*xsasak01@vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Extended Abstract

For an average customer, centralized exchanges are now the standard way to enter the world of digital money. Recently, with the increasing rate of insider attacks and exploits, users are increasingly losing trust in centralized exchanges. The recent attack within the FTX [1] exchange unleashed a wave of questions about transparency between the customer and the exchange. Is it possible to have a centralized exchange, where the operator does not have the complete power? Using the principles of trusted computing, we can introduce a secure gap between the power of the operator and the exchange.

We propose a proof-of-concept design for a centralized exchange that utilizes Intel SGX enclaves to create a secure gap between the operator and the exchange. Our design aims to solve the exchange non-equivocation and secure handling of exchange secrets, such as private keys. We benchmark and evaluate whether this solution is viable in a real production environment, since Intel SGX enclaves come with specific trade-offs, and the security protocols also introduce some form of overhead.

One of the existing proposed solutions called Tesseract [2] aims to solve a similar problem of centralized exchanges. The difference between our solution is that the Tesseract solution proposes a complicated time-locked deposit exchange method, which leads to the need for periodic cross-chain settlement transactions on blockchains. Periodic cross-chain swaps introduce potential timing bugs and require larger synchronization of exchange with the blockchain. Each settlement transaction on every blockchain costs a fee, which is another disadvantage.

Our exchange solution is made up of ledger $\mathbb{L}$ (history tree [3]), account state (Merkle-Patricia tree [4]) $\mathbb{A}$, and an indexed database $\mathbb{I}$. All these components are maintained only by the enclave and are completely outside the reach of the operator $\mathbb{O}$.

The ledger $\mathbb{L}$ is made up of exchange microblocks and uses a similar approach as the centralized blockchain Aquareum [5]. A single microblock contains a Merkle tree of microtransactions executed in a specific time window within the enclave, and these microtransactions change the state of accounts within the account state $\mathbb{A}$. A microtransaction can be one of these three types: deposit, bid, and withdrawal. Exchange can provide a signed membership proof of the microtransaction being in the microblock. Since the ledger is a history tree, exchange is able to provide signed incremental proof that the specific microblock hash is a member of the tree and that it commits the same history as the current microblock. The root hash of the account state is also included in the microblock, providing a snapshot of the account state as part of the microblock.

To achieve non-equivocation, the enclave deploys and maintains a smart contract $\mathbb{S}$ on the public blockchain. The contract is initialized and stores the following, the public key of the operator on the public blockchain $PK_{\mathbb{O}}^{PB}$, public key of the exchange enclave on the public blockchain $PK_{\mathbb{E}}^{PB}$, public key of the exchange enclave for the SGX TEE scheme $PK_{\mathbb{E}}^{TEE}$, exchange DNS or IP address $A_{\mathbb{E}}$ and the latest version and its root hash of the exchange ledger $H_{\mathbb{L}}$. This contract serves as an indisputable proof that the exchange enclave reached this state. The contract allows only the exchange enclave to update the latest ledger version, and the enclave does this periodically. In a real environment, a reasonable period must be found for updating the contract root hash. Since every update costs the operator a fee, there needs to be a compromise between speed and cost.

The exchange uses a decentralized identifier (DID) combined with verifiable credentials for user control. If the customer wants to open a new exchange account, he obtains a DID which includes his generated private-public key pair, that will be used for authentication on the exchange.

To open a new account the customer performs SGX remote attestation of the exchange, requests the enclave deposit address for a specific coin, and sends funds to the address. After transaction finalization, the customer sends the membership proof that the transaction is included in the specified block. Exchange validates all signatures and verifies the proof through enclave's internal SPV client. If everything is correct, the enclave opens an account and credits the customer's deposited funds into the account state of exchange.

In terms of bidding, every bid is matched, executed, and resolved virtually in the enclave. The customer submits a bid by providing the amount of specific coin that he wants to sell, followed by the amount of specific coin that he wants to buy. The enclave then attempts to match (fully or partially) the oposite bids, sorted by the cheapest, and if they exist, exchange performs the swap between the accounts by subtracting and adding balances.

Since trading and swapping occurs only virtually in the enclave, we provide a withdrawal action for the customer to retrieve his swapped coins. The customer submits a withdrawal request with the specified coin, amount, and address that is placed in the withdrawal queue. Enclave periodically submits blockchain transactions with multiple outputs, where each output represents a single withdrawal, and credits the specified amount to the customer's wallet. After the transaction is burried under enough blocks, the enclave confirms the withdrawal inside the enclave by subtracting the amount withdrawed from the frozen amount.

Compared to Tesseract [2], our solution does not require time-locked deposits implemented in blockchain transactions. More importantly, there is no need for settlement transactions to be resolved before the end of every time-locked deposit, and thus no need for an atomic cross-chain settlement protocol. We trade the ability of the customer to recover his funds after exchange disappearance for a more straightforward design. Keep in mind that if a potential breach of the enclave occurred, this time-locking deposit approach would still not help the customer protect his funds, since the time-locked deposit transaction output is only redeemable for exchange during the time period of the lock. If the breach could lead to the malicious operator obtaining the private keys created by the enclave for wallets, the operator would be able to redeem all time-locked deposits and steal the funds. In addition, our design provides a way for the customer to validate his own state of the account in the account trie and his microtransaction being in a microblock by using the smart contract $\mathbb{S}$.

Our proof-of-concept implementation is capable of handling around **35 customer deposits per second** and around **23 bids per second for single coin-pair**. There are two main sources of bottleneck. The first bottleneck comes from the fact that the Merkle-Patricia tree cannot be concurrently edited and recalculated, which means that the request for deposit or bid resolution requires exclusive access. The second source of bottleneck is the need for an exclusive lock for bid resolution for a single coin pair. The implementation currently supports Bitcoin and Litecoin and an exchange smart contract was implemented for Ethereum blockchain.

We proposed a proof-of-concept design of a secure exchange using Intel SGX. There is still much room for improvement, especially for bid resolving and account storing.

Acknowledgements

I would like to thank my supervisor Ing. Ivan Homoliak Ph.D. for his expertise on this topic.

References

[1] Nikhilesh De. Ftx files for bankruptcy protection in us; ceo bankman-fried resigns. *Coindesk*, November 2022.

[2] Iddo Bentov, Yan Ji, Fan Zhang, Lorenz Breidenbach, Philip Daian, and Ari Juels. Tesseract: Real-time cryptocurrency exchange using trusted hardware. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 1521–1538, New York, NY, USA, 2019. Association for Computing Machinery.

[3] Scott A Crosby and Dan S Wallach. Efficient data structures for tamper-evident logging. In *USENIX security symposium*, pages 317–334, 2009.

[4] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[5] Ivan Homoliak and Pawel Szalachowski. Aquareum: A centralized ledger enhanced with blockchain and trusted computing, 2020.