Tesseract: Real-Time Cryptocurrency Exchange Using Trusted Hardware

Iddo Bentov
Cornell Tech

Yan Ji
Cornell Tech

Fan Zhang
Cornell Tech

Lorenz Breidenbach
ETH Zürich and Cornell Tech

Philip Daian
Cornell Tech

Ari Juels
Cornell Tech

ABSTRACT

We propose Tesseract, a secure real-time cryptocurrency exchange service. Existing centralized exchange designs are vulnerable to theft of funds, while decentralized exchanges cannot offer real-time cross-chain trades. All currently deployed exchanges are also vulnerable to frontrunning attacks. Tesseract overcomes these flaws and achieves a best-of-both-worlds design by using a trusted execution environment. The task of committing the recent trade data to independent cryptocurrency systems presents an all-or-nothing fairness problem, to which we present ideal theoretical solutions, as well as practical solutions. Tesseract supports not only real-time cross-chain cryptocurrency trades, but also secure tokenization of assets pegged to cryptocurrencies. For instance, Tesseract-tokenized bitcoins can circulate on the Ethereum blockchain for use in smart contracts. We provide a demo implementation of Tesseract that supports Bitcoin, Ethereum, and similar cryptocurrencies.

不仅支持数字货币的交易，还支持代币的交易

精心制作

KEYWORDS

Cryptocurrency Exchanges; Frontrunning; Trusted Hardware

ACM Reference Format:
Iddo Bentov, Yan Ji, Fan Zhang, Lorenz Breidenbach, Philip Daian, and Ari Juels. 2019. Tesseract: Real-Time Cryptocurrency Exchange Using Trusted Hardware. In *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19), November 11–15, 2019, London, United Kingdom.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3319535.3363221

1 INTRODUCTION

The rise of Bitcoin [81] has spawned many hundreds of other cryptocurrencies as well as application-specific units of value known as crypto "tokens." This diverse ecosystem of assets has in turn led to a large and dynamic array of cryptocurrency *exchanges*, platforms that allow users to trade different cryptocurrencies against one another and/or for fiat currencies. At the time of writing, the aggregate daily trading volume of cryptocurrency exchanges exceeds $10 billion.

超过

Unfortunately, cryptocurrency exchanges suffer from a variety of security problems. Currently, the most popular exchanges are *centralized*, meaning that they hold traders' assets while trades are executed. Such exchanges support real-time trading of various cryptocurrencies and fiat currencies. They are vulnerable, however, to *theft* of traders' funds (cf. Appendix A.1). In a number of high-profile incidents, funds have been stolen when exchanges were breached or other forms of malfeasance took place [21, 46, 67, 76].

法定货币

Permissionless blockchains, however, are designed specifically to eliminate trust assumptions between transacting parties by avoiding centralization. A trust-free cryptocurrency exchange can be realized for transactions across such blockchains in the form of atomic intra-chain or cross-chain swaps (ACCSs) [31, 59], transactions that exchange cryptocurrencies between pairs of users in a fair, all-or-nothing manner. ACCSs, though, require users to wait many minutes (in fact, often hours) for a trade to execute. Additionally, atomic swaps in general aren't sufficient to realize an exchange: a mechanism for matching orders or otherwise performing price discovery is also necessary. Since ACCSs serve as a useful reference point, we elaborate on the concept and its limitations in Appendix B.

消除

原子交换不足以实现一个交易所，需要进行交易的匹配或者是价位的发现

The systemic risk of theft in centralized exchanges has led to the rising popularity of *decentralized* exchanges such as EtherDelta [92], 0x [107], and Kyber Network [70]. These systems hold traders' funds and settle transactions in smart contracts, eliminating the risk of theft in centralized exchanges. Unfortunately, they have other drawbacks. Their on-chain settlement means that they cannot support real-time trading. Moreover, while their use of smart contracts conveys an appearance of trustworthiness, they are vulnerable to various frontrunning attacks by miners and other users [40].

rising popularity of

消除

去中心化的交易所，没有办法做到实时交易

Achieving the best of both worlds has been a standing challenge, but a seemingly elusive one. An ideal cryptocurrency exchange would be *real-time* like a centralized exchange, meaning that participants can respond to price fluctuations and alter their positions with low latency. It would support even traders that utilize automated programs for high frequency trading and arbitrage (cf. [22]), who may wish to modify their positions in fractions of a second. At the same time, such an exchange would be *trust-free*, protecting against theft in the way that decentralized exchanges do, but also eliminating frontrunning attacks that exploit blockchain latencies.

变化

套利，仲裁。

消除，拍出

In this work, we present Tesseract, a cryptocurrency exchange that achieves this ideal set of properties. Tesseract is *real time*. Traders can rapidly observe the alterations in the buy (a.k.a. "bid") and sell (a.k.a. "ask") orders on the exchange, as well as external events (e.g., [112]), then modify their trading positions in milliseconds. By performing fast *price discovery*, they can drive price convergence so that the gap (a.k.a. "spread") between bids and asks is small, leading to efficient markets like those in major financial systems. Tesseract also prevents theft of users' funds by exchange operators and hackers as well as a variety of frontrunning attacks present in centralized and decentralized exchanges.

更改

集合，聚合

CCS '19, November 11–15, 2019, London, United Kingdom
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6747-9/19/11...$15.00
https://doi.org/10.1145/3319535.3363221

Tesseract supports *cross-chain* trading in which assets are exchanged across distinct blockchains. Trades within a single blockchain, e.g., exchange of tokens and Ether within Ethereum, can also be important (cf. [36, 37, 88, 93]). While this use case can be achieved at least in part using smart contracts, a significantly simplified variant of Tesseract can offer the added benefit of real-time trading, which smart contracts cannot support. Tesseract also supports a tokenization scheme that allows pegged tokens to circulate across blockchains, without relying on a human element for security (see Appendix E).

支持，

Tesseract relies on a trusted execution environment (TEE, cf. [87, 115]). This technology allows applications to execute within a protected environment called an *enclave*, that ensures confidentiality and software integrity. It enables Tesseract to behave like a trusted third party, controlling funds without exposing them to theft while preventing frontrunning by the exchange operator. Our security and trust assumptions are quite conservative, cf. Section 2.1.

Our reference implementation is built using SGX, which provides a TEE via an instruction-set architecture extension in recent-model Intel CPUs [14, 60, 61, 75]. While side-channel attackss [110] on SGX have been demonstrated, prominently Foreshadow [106] (that was later patched [39]), TEE technologies evolve as well. In particular, the Keystone project [62] is developing an open-source TEE.

The main challenge in the design of Tesseract is dealing with powerful network adversaries. Such adversaries can perform an *eclipse attack* in which an exchange is presented with fake blockchain data. We show how to address this problem by checkpointing trustworthy blocks within the Tesseract application and having it monitor the cumulative difficulty of newly furnished blocks. A network adversary can also suppress messages / transactions issued by the exchange in an attempt to interfere in on-chain settlement of trades, e.g., permitting partial settlement in which cryptocurrency flows to the adversary from a counterparty but not from the adversary, resulting in the adversary stealing funds. We express a theoretical solution to these network attacks in terms of an ideal functionality called a *refundable multi-input transaction* (RMIT). RMIT provides a conceptual springboard for securely architecting a secure cross-chain exchange. We present a highly efficient realization of RMIT in Tesseract, via a protocol that involves a network of TEE-backed nodes (Section 3.3, with an extended Paxos-based protocol in [27]). While only one node handles assets directly, others can execute or cancel transactions should the main node fail. This protocol enforces a key fairness property we define called *all-or-nothing settlement*.

Furnished 供应的

suppress 镇压，拦截

只有一个节点直接处理资产信息，其他节点在主节点发生故障之后，可以执行或者取消事务

In summary, our contributions in this paper are as follows:

- We introduce Tesseract, an TEE-backed cryptocurrency exchange that can support a wide variety of transaction types, with real-time cross-chain trading as its primary application.
- We consider powerful network adversaries that may seek to mount eclipse attacks or suppress transactions to achieve unfair settlement and thus theft of funds. We define a key fairness property called *all-or-nothing settlement* and show how to realize an exchange that achieves this property using as a conceptual building block an ideal functionality called RMIT.
- We present theoretical and practical techniques to achieve all-or-nothing settlement in Tesseract. The practical techniques include within-enclave blockchain monitoring to prevent eclipse attacks and use of a consensus group of TEE-backed nodes that can enforce and/or cancel transactions in the case that the main (asset-holding) exchange node becomes unavailable.
- We implement proof-of-concept of Tesseract, describing our parameter and design choices.

TEE，支持多种交易的类型，支持实时交易

为抵抗攻击者设计了一个公平属性 all–or–nothing settlement

为了属性设计了理论和时间的方案

在主交易节点失效的时候，有一个group的节点可以enforce 湖畔这cancel 交易

我们实现了这个概念

2 THE TESSERACT DESIGN

In this section we first specify our assumptions and then present an overview of the operation of Tesseract, describing how it achieves its security and performance goals. Specifically, Section 2.3 presents defense against powerful network adversaries that can eclipse the host; Section 2.4 gives the mechanism that prevents malicious administrators from mounting frontrunning attacks; Section 2.5 gives a defense-in-depth mitigation to TEE attestation failures.

抵抗eclipses 攻击
抵抗frontrunning 攻击

mounting 组织安排 defense-in-depth 深度防御

2.1 Threat Model

The Tesseract exchange achieves its security and performance goals by relying on a *trusted execution environment* (TEE), i.e., a hardware architecture that enables code execution in an isolated, tamper-free environment. The TEE can also attest [61] that an output represents the result of such an execution, and allows remote users to make sure that the attestation is correct. The *remote attestation* feature is essential for Tesseract, for reasons that will soon become clear.

We assume a strong network adversary (potentially the exchange operator) that can gain complete physical access to the host in which the funds are stored, giving her complete control of the operating system and network connections. We do assume that the code that runs inside the TEE enclave can neither be observed nor tampered with. Our reference implementation minimizes the risk of side-channel attacks by using constant-time and constant-memory code [109] for the critical part of Tesseract. In our threat model, the adversary's goal is to maximize her profit: she may directly attack the exchange (e.g., to attempt to extract secret keys that control the funds), but may also attack the network between users and the exchange to mount frontrunning attacks.

In a sense, the Tesseract exchange still relies on a trusted party in the form of the hardware manufacturer, because the attestation key inside CPU (and generates signatures for remote attestation) is provisioned by the manufacturer. It can be argued that a weaker yet similar form of trust is required in a practical instantiation of any cryptographic protocol, since the manufacturer may be able to attack the protocol by embedding malicious logic into the hardware. We critique this argument in Section 2.5, where we also give a double attestation scheme that makes Tesseract strictly more secure than exchange platforms that rely on centralized servers with no TEE. Thus, Tesseract still requires trust, but to a significantly lesser degree than centralized exchanges and other real-time exchange schemes (cf. Appendix A).

provision
准备，供给。。
需求

符号，记号， notation

2.2 Overview of Tesseract

Let us describe the operation of Tesseract, illustrated in Figure 1. For ease of notation, we use Bitcoin and Litecoin as the exemplary cryptocurrencies. We discuss more technical details in Appendix D.

Essentially, the Tesseract enclave is running light (a.k.a. SPV) blockchain clients. The enclave code is hardcoded with the hash

Enclave 跑的是一个轻客户端

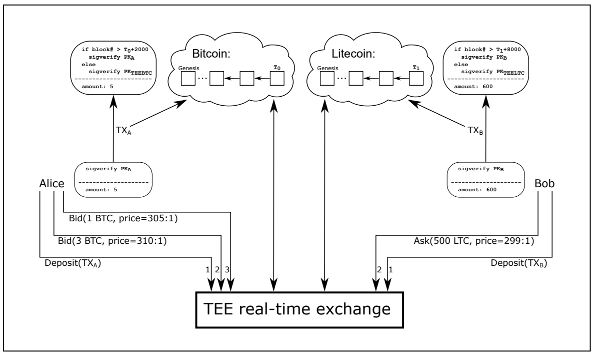


Figure 1: Illustrating deposits followed by bids/asks.

其中的code包含 genesis block， enclave 会同步最新的区块头，并验证区块头

of the Bitcoin genesis block, or a more recent "checkpoint" block of the Bitcoin blockchain. When the execution starts, the enclave receives the latest block headers from an untrusted Bitcoin client that runs on the same server machine. Each header is validated according to the protocol rule of the underlying cryptocurrency, specifically for Bitcoin the proof-of-work (PoW) in the header is validated against the current difficulty level. Each valid block is then added to a FIFO queue that is stored inside the enclave, where the size of the queue is set according to a parameter that specifies the maximum time window that the enclave maintains. The enclave maintains the same kind of queue for every other cryptocurrency that is supported by the Tesseract exchange service.

在初始化之后，会调用秘钥生成算法为每个支持的currency生成公私钥

连接

这些东西连接增加信息熵。
减少了私钥被攻击者知晓的概率

After initialization, the enclave invokes a key generation procedure to create a keypair (sk, pk) for each supported cryptocurrency. The randomness that we feed to the key generator is obtained by concatenating several sources: an hardware-based randomness instruction (RDRAND with SGX), the hashes of the latest blockchain blocks, OS provided randomness (via /dev/random), and the semi-trusted hardware clock (cf. Section 2.3). Each of these sources increases the entropy of the random data, and by combining them securely (via concatenation or hashing [84]) inside the enclave we reduce the likelihood that an adversary will have knowledge of the secret key sk.

其中，对于每个参与的数字货币区块链，其公钥会作为address

The enclave will then attest that a public key pk is its deposit address, for each cryptocurrency. The attestation to these public keys should be published through multiple services (such as websites, IPFS [25], and even Bitcoin and other blockchains). Our multi-server design (cf. Section 3.3 and [27]) also helps to make the attested deposit addresses publicly known. Figure 1 shows the two deposit addresses PK_{TEEBTC}, PK_{TEELTC}, for Bitcoin and Litecoin.

如果要开一个Teeseract账户，需要向交易所的账户充值。充值交易成功之后，这user将把成功的证明发送给tee
其中，这个证明中包含这个交易以及认证路径。

credit 信任

When a new user wishes to open a Tesseract account, she first needs to deposit a significant enough amount into a deposit address of the exchange. After the deposit transaction is confirmed on the blockchain, the (GUI client of the) user will transform the confirmed deposit into evidence that will be sent to the enclave. This evidence consists of the transaction that spends the coins into a deposit address of Tesseract, as well as an authentication path that consists of the sibling nodes in the Merkle tree whose root is stored in a block header, and the index of that block. Tesseract will credit the user's account (in the enclave) after verifying that the deposit transaction is valid, that the block B that contains the deposit belongs to the enclave's headers queue, and that B is buried under enough additional confirmations (see Section 2.3 for security analysis). Tesseract also protects against replay attacks, by requiring strictly increasing block indices for the user's deposits. In Figure 1, the evidence that Alice provides is Deposit(TX_A).

严格要求区块递增防止重放攻击

每一个充值都附加一个时间限定，在限定时间内，只有enclave可以花，超过限定时间，用户就有权限去拿回自己的这笔钱

这个机制同时保证了TEE无法提供服务的时候，用户可以拿回自己的钱

As shown in Figure 1, the output of a valid deposit transaction needs to specify a time limit (e.g., two weeks). Before the limit is reached, only the enclave can spend the deposit amount (for a Bitcoin deposit, this public key PK_{TEEBTC} is hardcoded in the output and the spending is done by creating a signature with the corresponding secret key SK_{TEEBTC}). After the time limit, the user can gain back control of her money by signing with a secret key that only she knows (see Appendix D for extra details). This deposit format ensures that the funds will safely be restored to the user if the Tesseract server becomes unavailable.

同时，当前区块里的难度也会在tee中硬编码，以防范攻击者用更小难度的区块欺骗

We note that the enclave is hardcoded with the current difficulty parameter of each PoW-based blockchain. At the beginning of the execution, the enclave will fetch blocks from genesis (or a more recent checkpoint), and verify that the chain reaches a block of the hardcoded difficulty level. This prevents an adversary (who has physical control of the Tesseract server) from feeding a low-difficulty fake chain to the enclave. The enclave updates the PoW difficulty level by inspecting the timestamps of block headers in the FIFO queue and applying the consensus rules of the cryptocurrency system (the queue size must be at least as the adjustment interval, which is 2016 for Bitcoin). This implies that an adversary cannot feed low-difficulty blocks to the enclave at a later time. The users of the Tesseract exchange can gain extra security by inspecting the latest block of each traded cryptocurrency and verifying (via remote attestation) that the enclave has the latest blocks, see Section 2.3 for details.

Malicious users may try to carry out a DoS attack on the Tesseract server by attempting to open many new accounts while providing fake deposits as evidence. Currently, Bitcoin blocks contain less than 4000 transactions, which implies that the authentication path requires 12 or fewer sibling nodes of the Merkle tree, and hence 12 invocations of a hash function. Thus, the time complexity of verifying the validity of a deposit is quite low. To further mitigate the prospects of a DoS attack, the enclave may require a moderate PoW done on the entire evidence data of the deposit (that the user will compute on her own), or simply limit the number of new account requests per timeframe.

One reason that the enclave maintains a queue of headers and fetches the additional block confirmations from the queue — as opposed to asking the user to concatenate the extra confirmations as part of the evidence of the deposit — is that the queue provides an undisputed point of reference in the form of the genesis (or checkpoint) block. That is to say, if there are two blockchains that use the same hash function for PoW and have a similar difficulty level, then a malicious user could deceive the enclave into accepting a deposit transaction that was confirmed on an incorrect blockchain. This approach also reduces the communication complexity between the Tesseract server and remote users. deceive 欺骗

After the user registers with Tesseract, her deposited amount is credited into her account entry in the array of users that is stored inside the enclave. Next, the user will be able to trade in real-time with other users who opened a Tesseract account, by sending

Table 1: Deposit confidence vs false positives

p	δ	n	Pr[Erlang(n, p) $\le \delta n$]	Pr[Erlang(n, 1) $> \delta n$]
$\frac{1}{10}$	2	60	2^{-75}	2^{-31}
$\frac{1}{10}$	2	120	2^{-145}	2^{-58}
$\frac{1}{5}$	1.5	120	2^{-92}	2^{-21}
$\frac{1}{4}$	1.3	120	2^{-82}	2^{-10}

通过安全通道把order发给tee

bid/ask orders to the Tesseract server via a secure channel (see Section 2.4). If the user wishes to deposit other currencies into her account, she can then send similar authentication paths as evidence.

此外，这个账户还支持存储其他的币到这个账户

In Figure 1, Bob opens an account with Deposit(TX$_{\mathrm{B}}$), and then asks to sell 500 LTC for the price of 299 LTC per BTC. Since Alice's bids are with a price of 305 LTC per BTC and higher, there is no match yet, and the requests of Alice and Bob are recorded in the order book kept inside the enclave. Each user can request her recent trading history via the secure channel, and cancel her pending orders. The Tesseract server publishes an anonymized version of the order book (i.e., price and volume of each order, without usernames) with remote attestation; hence anyone can observe the price spread of the exchange. Since order book updates can occur at a very rapid rate, we reduce the amount of TEE attestations via delayed randomized checkpoints: the enclave always outputs the anonymized order book without a signature, and outputs a delayed attestation (that include an incremental counter) only for randomly selected data points. The administrator of the Tesseract server provides her part of the double attestation for all the data points (using HTTPS, see Section 2.5). Thus, an administrator that publishes fake order book data repeatedly will (w.h.p.) be detected. The administrator still has a potential advantage over all other traders because she is the first to see each order book, but the advantage is quite small. E.g., if Alice sends a buy order (via TLS, cf. Section 2.4) with a typo, the sell orders (in the enclave's order book) that match her order will execute before the enclave outputs the next order book (the administrator stands to gain if the order book is shallow or empty).

Order book更新的速度较快，为防止其速度受到远程认证的影响，我们使用delay认证的方式

Real-time trading among the users will cause frequent updates to the balances of their accounts inside the enclave, but these updates are not reflected on the actual cryptocurrency systems yet. If nothing else were to happen, the entire process would just be a sandbox or playground, as the users will simply claim their original money after the time limit of their deposits is reached. Therefore, from time to time (e.g., once a day) Tesseract will broadcast to the cryptocurrency networks "settlement" transactions that commit the current account balances of the users. See Figure 3 for an illustration, and Section 3 regarding a secure settlement protocol.

快速的更新

The enclave extends the time limit of each user's output in the settlement transactions that it constructs (e.g., if the user could control the output in 5 days before the settlement, then she could control the output in 19 days after the settlement). This allows uninterrupted trading by active traders. To minimize the size of the settlement transactions, users who did not trade are not included in the inputs and outputs. When some of a user's funds are in an output whose time limit is about to expire, the user will be prohibited from trading. The user is permitted to send a renewal request *before* the expiration, in case she was unlucky and none of her trade orders were matched (renewal after the expiration can be exploited by malicious users who would create conflicting transactions near the time limit). The user can also request an early withdrawal of some of her funds. This is done by directing the enclave to prepare an output that is controlled only by the user, in the next settlement.

禁止

The Tesseract exchange collects a proportional fee for each successful trade (e.g., 0.1% from both ends of a trade), and a flat fee for early withdrawal and renewal requests. The enclave requires each user to have a minimal amount of funds at all times, and limits the total number of pending orders that a user may have in the order book – users who flood the exchange with an excessive number of orders may be penalized (by confiscating some of their funds) and blacklisted for a period of time. The fees that Tesseract collects are needed in order to pay miner fees for the settlement transactions.

2.3 Eclipse Attacks

We assume an adversary $\mathcal{A}$ that controls $p < \frac{1}{2}$ fraction of the computational power of a blockchain that the enclave interacts with, and also has physical access to the Tesseract server. Thus, $\mathcal{A}$ can cut the communication between the enclave and the network, and feed the enclave fake blocks.

Assuming a naive enclave implementation, $\mathcal{A}$ can mount an *Eclipse attack* [58] as the following example illustrates: $\mathcal{A}$ cuts the enclave off from the Bitcoin network and presents it with a fake blockchain containing a deposit transaction TX$_{\mathrm{fake}}$. As a result, the enclave credits $\mathcal{A}$ with a higher Bitcoin balance, which $\mathcal{A}$ trades for Litecoin inside the enclave. When the enclave publishes the next settlement transactions on the two blockchains, $\mathcal{A}$ will have traded her fake Bitcoin for real Litecoin: The Bitcoin settlement transaction will not be valid because it spends an output from TX$_{\mathrm{fake}}$ which was never included in the real Bitcoin blockchain. However, the Litecoin settlement transaction *will be valid*, resulting in $\mathcal{A}$ profiting.

To defend against this attack, we rely on the fact that the rate at which $\mathcal{A}$ can feed fake blocks to the enclave is at least twice slower than in the absence of an attack. (Since $p < \frac{1}{2}$.) Assuming that the TEE has a trusted clock[1], the enclave can impose a rule that requires waiting for additional confirmations if the blocks arrive too slowly. We note that the Tesseract enclave is assumed to be running continuously, since our enclave code disallows rollbacks [72, 101] by design (cf. Section 3.3 and [27] regarding our approach to resiliency).

The time between every two consecutive Bitcoin blocks is an exponentially distributed random variable. Hence, for a rule that dictates whether blocks arrive too slowly we should consider the sum of exponential random variables, known as the Erlang distribution. Let n be the number of blocks that a deposit needs to be buried under before it is credited by the enclave. Let δ be the multiplicative slowness factor by which blocks are allowed to arrive. E.g., $\delta = 3$ means that blocks that arrive 3 times slower than the expected time (or more slowly than that) will trigger the enclave to wait for n extra block confirmations before accepting any deposits.

Setting δ to a high value reduces the probability of a false positive (i.e., a rejected deposit when no attack is taking place and the honest chain growth was unluckily slow during some timeframe). However, a high δ also increases the prospects of an attack. For any $\delta > 1$, it is possible to set a large enough n so that the probability of a successful attack becomes negligible. However, a large n implies that honest

[1] The trusted *relative* timer that SGX can provide is adequate, see [1].

Order Book (BTC/USD)

Buying		Selling	
Price	Volume	Price	Volume
\$850	2	\$890	3
\$840	5	\$906	5
\$820	5	\$945	4

Arrival of new orders:

```
1. Alice:  buy($870, 10)
2.   Bob: sell($820, 10)
```

Frontrunning:

```
1. Adversary:  buy($851, 10)
2.       Bob: sell($820, 10)
3.     Alice:  buy($870, 10)
4. Adversary: sell($870, 10)
```

Figure 2: Example of frontrunning.

users need to wait for a long time before their deposit is confirmed, which makes the Tesseract exchange service unattractive.

In Table 1 we provide exemplary concrete parameters for n and δ. E.g., the third row of Table 1 shows that with $n = 120$ (20 hours on average in Bitcoin) and $\delta = 1.5$:

- An adversary with computational power $p \leq \frac{1}{5}$ can mount a successful eclipse attack on the enclave with probability 2^{-92} or smaller.
- In expectation, an honest user will need to wait for extra confirmations once in every $\approx$ 2 million deposits that she makes.

While the concrete parameters that can be obtained are already quite reasonable, let us stress that prudent users of the Tesseract exchange will not be exposed to eclipse attacks at all. Any user can simply compare the latest blocks in the actual cryptocurrency networks with the latest blocks that Tesseract enclave publishes (with remote attestation), and cancel her bids/asks in case of a discrepancy. In the example above, the honest P_j will avoid P_i's attack by observing that the latest Bitcoin blocks that Tesseract published are inconsistent with the real Bitcoin network, and refuse to trade her LTC for BTC. Our practical instantiation of Tesseract has another layer of security that further protects (incautious) users from eclipse attacks, see Section 3.3.

2.4 Secure Communication

For each user who has already opened an account with Tesseract, we establish a secure channel (e.g., TLS) when the user wishes to communicate with the enclave. The reasons for a channel with authenticated encryption are:

- *Fast identification:* The authenticated messages in the TLS Record Protocol are computed via symmetric-key operations, after the initial key exchange (done via public-key operations in the Handshake Protocol) to establish the channel. Since symmetric-key operations are an order of magnitude faster than public-key operations, a persistent TLS connection delivers performance suitable for real-time trades.
- *Frontrunning prevention:* An adversary can try to inspect the entire communication flow that arrives at the Tesseract server, learn information regarding real-time actions of other users, and perform trades that exploit this information. Encrypted communication avoids such attacks.

An example of a frontrunning attack is shown in Figure 2. There, Alice believes that the BTC price is going to rise. Therefore, she places an order to buy 10 BTC at \$870 each, so that any of the current sellers will match her order first. On the other hand, Bob believes that the price of BTC is going to drop, and he therefore places an order to sell his 10 BTC for a price that is as low as \$820. Given the public order book, Bob's intention is thus to sell 2 BTC for \$850, 5 BTC for \$840, and 3 BTC for \$820. If the trades are executed in this order, it will be to the benefit of Bob, because he will actually sell 10 BTC to Alice for \$870 each. However, an adversary with this knowledge can permute the orders and insert her own new orders. In this scenario, the adversary would be guaranteed to gain $\$10 \cdot (870 - 851) = \$190$, by buying Bob's 10 BTC cheaply and then selling it to Alice.

Since all users send encrypted messages through their secure channels, an adversary with a physical control of the Tesseract server cannot frontrun other users. To the best of our knowledge, all other designs of real-time cryptocurrency exchanges are exposed to these kinds of frontrunning attacks. Non-real-time exchanges such as TEX [63] prevent frontrunning attacks, by employing time-lock puzzles and progressing in delayed batches.

We note that an adversary may still observe patterns of communication at the IP-level and try to learn information about the traders. An IP-level anonymizer (e.g., Tor [45]) is inapplicable as a mitigation technique against such adversaries, since the extra latency [80] that Tor users incur will put them at a disadvantage relative to non-Tor users that engage in real-time trading. As an alternative, the user's client can randomly inject dummy data into the TLS channel (which would be ignored on arrival), thereby making it more difficult to track communication patterns. Furthermore, in future versions of Tesseract we plan to allow users to upload an algorithmic trading program to their enclave account (for a fee), that will enable them to issue multiple trading orders without communication with the server. The use of automated trading programs is quite popular in centralized exchanges (cf. [22]), although these automated traders do communicate each of their orders to the server.

2.5 Double Attestation

Several reputable providers may wish to offer different variants of the Tesseract service (perhaps with their own tokenized coins and other digital assets, cf. Appendix E). This raises the following question: does a single entity (i.e., the hardware manufacturer) have the power to compromise the security of all the Tesseract-based platforms, simultaneously?

No such single entity exists with regard to centralized exchanges (cf. Appendix A.1), because these exchanges are independent of one another. That is to say, a security breach of one centralized exchange will not have a direct impact on the users of the other centralized exchanges.

For trusted hardware with remote attestation support, the plain way that the manufacturer can break security is by attesting to fraudulent data. In our context, suppose for example that there are two Tesseract-based exchanges X_1, X_2 that invite users to deposit their funds to $\mathrm{PK}_{\mathrm{TEEBTC1}}$ and $\mathrm{PK}_{\mathrm{TEEBTC2}}$, respectively. If Intel has knowledge of the secret signing keys sk_1, sk_2 that are embedded into the CPUs of X_1 and X_2, then it can forge signatures that attest to fresh ephemeral public keys $\mathrm{PK}'_{\mathrm{TEEBTC1}}, \mathrm{PK}'_{\mathrm{TEEBTC2}}$ that Intel would generate together with the corresponding secret keys $\mathrm{SK}'_{\mathrm{TEEBTC1}}, \mathrm{SK}'_{\mathrm{TEEBTC2}}$. Thus, Intel will be able deceive users into sending their deposits to $\mathrm{PK}'_{\mathrm{TEEBTC1}}, \mathrm{PK}'_{\mathrm{TEEBTC2}}$, and then steal funds that users wished to deposit to X_1, X_2.

The manufacturer may also break security by embedding malicious logic into the hardware. For instance, whenever an application executes code that generates a (supposedly) random secret key, the key will actually be generated in a way that can be predicted by the manufacturer. While this attack would be easy enough if there were one assembly opcode that generates a random key (the malicious opcode can use a randomness source with low entropy), it is far more difficult to achieve predictable behavior for any application-level code that is executed by a general-purpose CPU.

Another attack vector that the hardware manufacturer may attempt is simply to send the data that a CPU generates over the network (to the manufacturer's address), without consent or knowledge of the administrator of the server computer. This is indeed a concern with Intel's Management Engine (see [91]), but it is not an inherent defect of the trusted hardware model (hopefully the Management Engine will allow opt-out).

Similarly to [97], the Tesseract platform protects against false remote attestation by attaching a secondary signature – created by the administrator of the platform – to the attested data. Following the above example, the users of X_1 (resp. X_2) will take into consideration the reputation of the administrator of X_1 (resp. X_2), and reject the attested data unless it was signed both by the TEE-enabled CPU and by the reputable administrator. Hence, the hardware manufacturer alone cannot attack all Tesseract-based exchanges, since the manufacturer has to collude with the administrator of an exchange in order to create a fraudulent attestation. This implies that Tesseract is strictly more secure than centralized exchanges.

The double attestation mechanism is also efficient, since the secondary signature is rarely needed. Specifically, the secondary signature is required only once for the hardware-associated public key identity (cf. [27, Section 6.1]) of the enclave, and this identity can then establish the TLS channel with each user. All further communication in a TLS channel (e.g., bid/ask orders) is done without attestation. For non-user-specific data such as real-time updates to the public order book, the secondary signature is already implicit if HTTPS is used to view this data.

3 ATOMIC CROSS-CHAIN SETTLEMENTS

Assume first that Tesseract only supports the trading of digital assets that circulate within a single cryptocurrency. In this case, the publication of each settlement transaction — that reflects the account balances of the users after trading in a time period — does not entail the risk of an adversary stealing funds from honest users. The reason is that an invalid deposit (see Section 2.3) or blockage of the settlement will amount just to a DoS attack, since all the users will claim their prior funds after the time limit in the output of their original deposit (or the last settlement transaction) expires.

On the other hand, trading among multiple cryptocurrency systems (that are independent of one another) may allow an adversary to steal funds from honest users. We provide an illustration of the risk in Figure 3. Suppose for instance that 1 BTC is worth $2000, and also that the market price of 1 BTC is 300 LTC. In the illustration, Alice and Bob traded 1000 BTC (i.e., $2 million worth of BTC) for 300000 LTC (i.e., $2 million worth of LTC), while Carol and Dave traded 1 BTC for 300 LTC. Thus, the enclave will construct and sign 对手，敌手 the Bitcoin and Litecoin settlement transactions, and attempt to broadcast the settlements to the Bitcoin and Litecoin networks. An adversary with physical access to the Tesseract server can collude 勾结，串谋 with Alice and intercept the Bitcoin settlement transaction when it leaves the CPU but before it is broadcast to the Bitcoin network, and let the Litecoin settlement transaction go through and reach the Litecoin network. The result is that the transfer of ownership of $2 million worth of LTC from Bob to Alice will be committed on the Litecoin system, while the transfer of ownership of $2 million worth of BTC will never occur. In effect, Bob lost $2 million worth of funds to Alice.

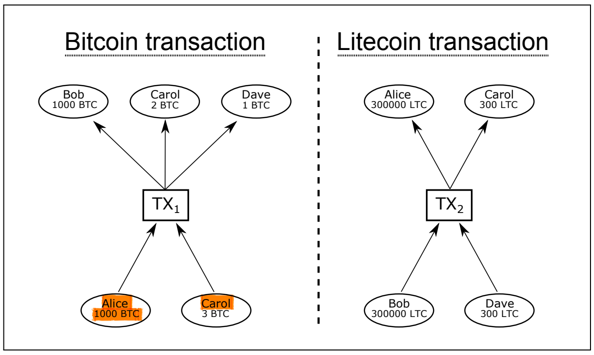


Figure 3: The cross-chain settlement problem.

Let us provide security definitions that capture the above fairness problem.

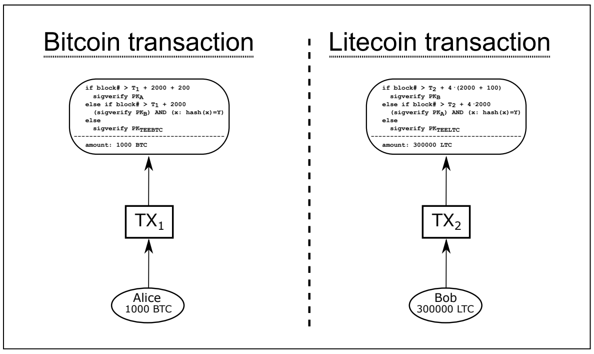
Definition 3.1 (All-or-nothing settlement). Given the transaction tx_1 for system C_A and the transaction tx_2 for system C_B, an all-or-nothing cross-chain settlement is a protocol that guarantees that

(1) Both tx_1 will become confirmed on system C_A and tx_2 will become confirmed on system C_B, or
(2) Neither tx_1 will become confirmed on system C_A nor will tx_2 become confirmed on system C_B.

In our context, C_A and C_B are cryptocurrencies. We stress that parties that execute the consensus protocol for C_A may be unaware of the existence of C_B, and vice versa. 反之亦然

Notice that Definition 3.1 does not imply that honest users are fully protected against financial loss. Specifically, an adversary $\mathcal{A}$ that prevents both tx_1 and tx_2 from being confirmed may benefit at the expense of honest users: $\mathcal{A}$ may wish to renege on a trade after observing some external events and/or price fluctuations that worked to her disadvantage. Still, Definition 3.1 implies better security than that of the commonplace centralized exchanges (cf. Appendix A.1), because the users of such centralized exchanges run not only the risk that their trades will be reversed but also the risk that their initial funds will be stolen.

不仅是交易被逆转，初始的资金也有可能丢失



```
if block# > T1 + 2000 + 200
  sigverify PKA
else if block# > T1 + 2000
  (sigverify PKB) AND (x: hash(x)=Y)
else
  sigverify PKTEEBTC
--------------------------------------
amount: 1000 BTC
```

```
if block# > T2 + 4·(2000 + 100)
  sigverify PKB
else if block# > T2 + 4·2000
  (sigverify PKA) AND (x: hash(x)=Y)
else
  sigverify PKTEELTC
--------------------------------------
amount: 300000 LTC
```

Figure 4: Settlement with two parties.

Definition 3.2 (Unprivileged settlement). Let $U_1^{\text{in}}, U_2^{\text{in}}$ denote the sets of users in the inputs of the transactions tx_1, tx_2, and let $U_1^{\text{out}}, U_2^{\text{out}}$ denote the sets of users in the outputs of tx_1, tx_2. Let $U = U_1^{\text{in}} \cup U_2^{\text{in}} \cup U_1^{\text{out}} \cup U_2^{\text{out}}$. An unprivileged cross-chain settlement is a protocol that satisfies Definition 3.1 in the presence of an adversary $\mathcal{A}$ who can obtain any information that every user $P \in U$ accesses, at the moment that the information was accessed.

交易方不能使用共享秘钥的方式

In essence, Definition 3.2 implies that honest traders cannot utilize secret data during the settlement protocol (such as picking a secret $x \in \{0,1\}^\lambda$ in the first step of the ACCS protocol in Appendix B), because $\mathcal{A}$ could break the security by gaining access to any sensitive data that honest traders attempt to use. Thus, Definition 3.2 captures a rushing adversary who has physical control over the TEE server and can intercept all the data that leaves the CPU, before honest users have an opportunity to make use of this data in a secure fashion. Note that Definition 3.2 does not permit $\mathcal{A}$ to observe the secret keys that enable honest users to spend their funds, as long as they do not access their secret keys during the settlement protocol.

In fact, Definition 3.2 gives $\mathcal{A}$ more power than a real-world adversary with physical control over the TEE server. Consider for instance a protocol where in the first step the enclave encrypts data using Carol's public key, and attempts to send the encrypted data to Carol over the network. In that case, $\mathcal{A}$ will not be able to obtain the data that Carol accesses; the only action available to $\mathcal{A}$ is to mount a DoS attack and not let the protocol make progress. The motivation for the more conservative definition is that we wish to support settlement transactions among a large number of users (e.g., thousands) and multiple cryptocurrency systems, where the users can be anonymous and can create Sybil accounts. In this setting, it is difficult to design a secure protocol that sends sensitive data to rational users (with the expectation that they will act in their own self-interest), due to the possibility of malicious coalitions with Sybils who would be willing to sacrifice some of their funds. For this reason, Definition 3.2 denies the enclave the power to communicate privately with individual users. 拒绝。否认

复杂的

Thus, intricate solutions to the all-or-nothing settlement problem are needed mainly because our goal is to support many anonymous traders. Let us in fact demonstrate that with a few users, the all-or-nothing settlement problem can become easy. In Figure 4, Alice and Bob again wish to trade \$2 million worth of BTC for LTC, but they are the only users of the Tesseract exchange. Here, the enclave prepares the settlement transactions TX_1, TX_2 that keep the enclave in control in the next two weeks (2000 blocks where T_1 is the head of the Bitcoin blockchain, and 8000 blocks where T_2 is the head of the Litecoin blockchain). This enables Alice and Bob to continue to trade, if they wish to. The secret data $x \in \{0,1\}^\lambda$ is generated inside the enclave. After the enclave receives evidence that TX_1 and TX_2 are both confirmed, it sends x in encrypted form *only* to Alice, over a secure channel. After the two weeks, the outputs can be redeemed using x, otherwise the timeouts allow the funds to be returned to each user. As with the ACCS protocol (cf. Appendix B), the timeout in TX_1 is longer, so Bob will have enough time to redeem the 1000 BTC after Alice reveals x, spending 300000 LTC.

如果有更少的user，他的清算问题会更简单

A和B是tee的唯一用户

准备清算，tee在接下来的两周都会受控制

秘密x在TEE里面生成，在收到tx1和tx2的确认信息后，将加密后的x发送给A通过隐私信道。

其输出可以使用x赎回，或者超时被退回

Protocol Π_{simp}

(1) The enclave picks a symmetric key $K \in \{0,1\}^\lambda$.
(2) The enclave embeds K into TX_1, TX_2.
(3) The enclave sends $ct = \text{encrypt}_K(\text{TX}_1, \text{TX}_2)$ to $S_1, S_2, \ldots, S_N$.
(4) The enclave waits for acknowledgements from $S_1, S_2, \ldots, S_N$.
(5) The enclave broadcasts TX_1 to C_1 and TX_2 to C_2.
(6) Each S_j that sees TX_i but not TX_{3-i} will fetch K from TX_i, decrypt ct, and broadcast TX_{3-i} to C_{3-i}.

Figure 5: Naive protocol for fair settlement.

Let us note that Definition 3.2 does not give $\mathcal{A}$ the power to observe secret information inside the enclave. In the Tesseract implementation, this is justified because we use a constant-time constant-memory library for cryptographic operations [109], reducing the potential for side-channels greatly.

We now present solutions to the all-or-nothing settlement problem, in a setting that involves many anonymous traders.

3.1 Naive Protocols

复杂的

To clarify why an intricate protocol is needed, we first describe a simple protocol Π_{simp} that relies on N extra servers $S_1, S_2, \ldots, S_N$ that are supposedly reputable. See Figure 5.

The cryptocurrency systems C_1 and C_2 can be for example Bitcoin and Litecoin as in Figure 3. The embedding of K into TX_1 and TX_2 can be done with the `OP_RETURN` script instruction [23], which allows storing arbitrary data on the blockchain as an unspendable output (for a small fee). It is not possible to mount a malleability attack that removes K from TX_1 or TX_2, because the signatures for TX_1 and TX_2 are over the entire transaction data (i.e., data that includes the `OP_RETURN` output).

Since information that is published on a blockchain becomes publicly available, the idea behind Π_{simp} is that any non-corrupt server S_i will be able to impose fairness by fetching K from a public blockchain and decrypting the ciphertext ct, because ct is already in S_i's possession. intercept 拦截

Unfortunately, Π_{simp} is insecure, due to a race condition. The adversary $\mathcal{A}$ can intercept both TX_1 and TX_2, but broadcast neither of them initially. Since the users' outputs must have a time limit (see Section 2), $\mathcal{A}$ will wait until an input (that belongs to a corrupt user P_j) in TX_i is about to expire, and then broadcast TX_{3-i}. Then, Functionality RMIT (refundable multi-input transaction)

Notation: let C be a cryptocurrency system.

Upon receiving $tx = (\{in_1, \ldots, in_k\}, \{out_1, \ldots, out_n\}, \phi_1, \phi_2)$

1. Verify $\forall j \in [k]:\ in_j$ is unspent in C.
   - If the verification failed then abort.
2. Verify $\sum_{j=1}^{k} \mathsf{amount}(in_j) \geq \sum_{j=1}^{n} \mathsf{amount}(out_j)$.
   - If the verification failed then abort.
3. Make $\{in_1, \ldots, in_k\}$ unspendable in C.
4. Wait to receive a witness w
   - (a) If $\phi_1(w) = 1$ then commit $\{out_1, \ldots, out_n\}$ to C, and terminate.
   - (b) If $\phi_2(w) = 1$ then make $\{in_1, \ldots, in_k\}$ spendable in C, and terminate.
   - (c) Otherwise, return to Step 4.

Figure 6: The ideal functionality RMIT.

$\mathcal{A}$ will instruct P_j to spend that input, thereby invalidating TX_i. Hence, even if all of the servers $S_1, S_2, \ldots, S_N$ are honest, they may not have enough time to fetch K from TX_{3-i} and broadcast the decrypted TX_i.

可以减轻tee之间的依赖

If the cryptocurrency systems C_1,C_2 allowed transactions to embed large arbitrary data, then it would have also been possible to eliminate the reliance on $S_1, S_2, \ldots, S_N$. Briefly, each TX_i will embed the TX_{3-i} data in a designated output, the enclave will 嵌入 broadcast both TX_1 and TX_2, and any user would then have the opportunity to enforce fairness. This would bloat C_i with the entire TX_{3-i} data, which is undesirable — there are risks associated with a popular decentralized cryptocurrency that allows embedding of large data (e.g., illegal content). In any event, this approach is insecure due to the same race condition that Π_{simp} exhibits.

In the following section, we give a theoretical protocol Π_{theo} that avoids the race condition, using scripts with PoW-based logic that ensures the occurrence of certain conditions on another blockchain.

我们提出一个理论的方案可以避免race condition，使用基于pow的原理证明另外的一个区块确定满足这个条件

3.2 Theoretical Protocol

详细描述的

我们标记第二个交易发生在第一个交易清算的结果下

Let us present a theoretical protocol for the all-or-nothing settlement problem, which solves the race condition that Section 3.1 elaborates upon. Following Section 3 and Figure 3, we condition the second settlement transaction TX_2 on the result of the first settlement transaction TX_1, by constraining TX_2 with PoW-based predicates that verify certain events' occurences on another blockchain. 断言，验证是否有确定的时间发生在另外的区块链上

As we will see, this approach is problematic with the current Bitcoin protocol. Thus, we first describe the settlement protocol in an hybrid world that has an ideal "refundable multi-input transaction" (RMIT) functionality, defined in Figure 6. 退还多输入的事务

The description of TX_1, TX_2 is outlined in Figure 7. We use the notation $\mathsf{TX}_{i,j}$ to denote that TX_i was updated by supplying w that satisfied ϕ_j. The secrets $x_1 \in \{0,1\}^\lambda, x_2 \in \{0,1\}^\lambda$ are generated inside the enclave. The predicates ϕ'_1, ϕ'_2 are specified in Figure 8. To elaborate, the hardcoded parameter D_0 specifies a difficulty level for PoW mining, ℓ_1 is an upper bound on the length of an authentication path of a Merkle tree, and ℓ_2 is a PoW confidence parameter. The input witness w for ϕ'_1 consists of up to ℓ_1 sibling hash values v_j in the authentication path (with direction $d_j \in$ {'L','R'}) for the leaf transaction y, together with exactly ℓ_2 block headers $H_1, H_2, \ldots, H_{\ell_2}$. The predicate ϕ'_1 will verify that $\mathsf{TX}_{1,1}$ is in a leaf that reaches some root value r, and that r is extended by valid proofs of work $H_1, H_2, \ldots, H_{\ell_2}$ that meet the difficulty level D_0. The input witness w for ϕ'_2 does the same, but also verifies that there is a valid PoW chain of at least ℓ_3 blocks between the hardcoded b_1 and $\mathsf{TX}_{1,2}$.

We describe the theoretical protocol Π_{theo} for all-or-nothing settlement in Figure 9. Note that the enclave constructs TX_2 only after it receives the evidence that TX_1 was confirmed in the end

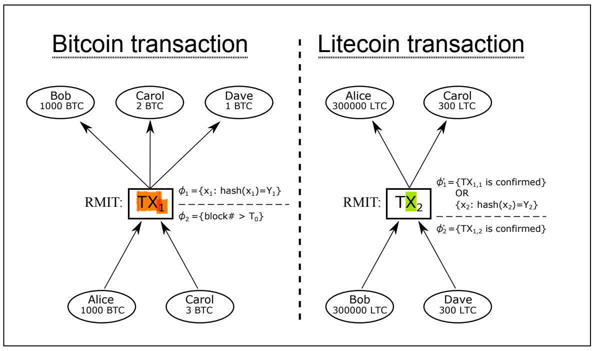


Figure 7: Theoretical fair settlement transactions.

Predicate ϕ'_1

Hardcoded parameters: $\mathsf{TX}_1, D_0, \ell_1, \ell_2$

Input: $w = ((v_1, d_1), (v_2, d_2), \ldots, (v_k, d_k), y, H_1, H_2, \ldots, H_{\ell_2})$

1. Embed $\mathsf{hash}(\mathsf{TX}_{1,1})$ into y
2. For $j = 1$ to $\min(k, \ell_1)$
   - If d_j ='L' then $y := \mathsf{hash}(y, v_j)$ else $y := \mathsf{hash}(v_j, y)$
3. For $j = 1$ to ℓ_2
   - Embed y into H_j and compute $y := \mathsf{hash}(H_j)$
   - If $y > D_0$ then return `false`
4. return `true`

Predicate ϕ'_2

Hardcoded parameters: $\mathsf{TX}_1, D_0, \ell_1, \ell_2, \ell_3, b_1$

Input: $w = (G_1, \ldots, G_n, (v_1, d_1), \ldots, (v_k, d_k), y, H_1, \ldots, H_{\ell_2})$

1. $z := b_1$
2. For $j = 1$ to $\max(n, \ell_3)$
   - Embed z into G_j and compute $z := \mathsf{hash}(G_j)$
   - If $z > D_0$ then return `false`
3. Embed $\mathsf{hash}(\mathsf{TX}_{1,2})$ into y
4. For $j = 1$ to $\min(k, \ell_1)$
   - If d_j ='L' then $y := \mathsf{hash}(y, v_j)$ else $y := \mathsf{hash}(v_j, y)$
5. Embed y into H_1 and compute $y := \mathsf{hash}(H_1)$
6. If $y \neq z$ then return `false`
7. For $j = 2$ to ℓ_2
   - Embed y into H_j and compute $y := \mathsf{hash}(H_j)$
   - If $y > D_0$ then return `false`
8. return `true`

Figure 8: The cryptocurrency scripts ϕ'_1, ϕ'_2.

Protocol Π_{theo}

(1) The enclave releases TX_1 and waits for evidence that it was confirmed on the cryptocurrency system C_1.
(2) The enclave releases TX_2 and waits for evidence that it was confirmed on the cryptocurrency system C_2.
(3) The enclave releases x_1 and waits for evidence that $\text{TX}_{1,1}$ was confirmed on the cryptocurrency system C_1.
(4) The enclave releases x_2.

Figure 9: Theoretical protocol for fair settlement.

of Step 1, by hardcoding b_1 as the hash of the block in which TX_1 resides.

Essentially, Π_{theo} avoids the race condition by first making sure that TX_1 was resolved on the cryptocurrency system C_1 either by committing the output or by committing the inputs, and then allowing TX_2 to commit accordingly in the cryptocurrency system C_2. If $\mathcal{A}$ carries out a DoS attack before x_1 is released in Step 3, then the users will gain possession of their inputs in the C_1 after block T_0 is reached (see Figure 7), which would be followed by the miners of C_1 starting to create a witness w that satisfies $\phi_2'(w) = 1$ and thus allowing users to gain possession of their inputs in C_2. If the enclave exposes x_1 in Step 3, it is still the case that the miners of C_1 will be harnessed to resolve TX_1 in one of the two possible ways.

In the case that no attack is taking place, the enclave will release x_2 in Step 4, thereby allowing the settlement to complete quickly and without asking the miners of C_2 to evaluate a complex condition that relates to another blockchain.

However, the assumption regarding the computational power of $\mathcal{A}$ has to be slightly less conservative in comparison to the power that is needed to mount a classical double-spending attack [94], because Π_{theo} enables $\mathcal{A}$ to gain a minor head start that depends on the parameter T_0. Specifically, $\mathcal{A}$ can intercept x_1 in Step 3 and use her own computational power (and x_1) to create a hidden chain w_1 that spends TX_1 into $\text{TX}_{1,1}$. The miners of C_1 will create the witness w_2 in which TX_1 is spent into $\text{TX}_{1,2}$, but they will only begin to work on w_2 after block T_0 is reached.

The success probability of an attack with a duration of T_1 blocks for the head start is

$$\sum_{k=0}^{\infty} \Big(\Pr[\text{NegBin}(T_1, p) = k] \cdot \Pr[\text{NegBin}(\ell_2, p) \geq \ell_2 - k]\Big).$$

The first negative binomial variable counts the number of blocks that $\mathcal{A}$ creates during the time that the honest miners are creating T_1 blocks. This corresponds to the head start, because these T_1 blocks will not contribute to the witness that the predicate ϕ_2' requires. The second negative binomial variable counts the number of blocks that $\mathcal{A}$ creates while the honest miners are creating ℓ_2 blocks. If $\mathcal{A}$ can extend her head start to reach ℓ_2 or more blocks before the honest miners, then the attack succeeds.

In Table 2, we give exemplary figures for the attack on Π_{theo}. For easy comparison, we also include the success probability without a head start (i.e., $T_1 = 0$), which is simply the probability $\Pr[\text{NegBin}(\ell_2, p) \geq \ell_2]$.

Table 2: Breaking the security of Π_{theo}

p	T_1	ℓ_2	with head start	with $T_1 = 0$
$\frac{1}{3}$	6	50	0.0016	0.0003
$\frac{1}{5}$	10	50	2^{-30}	2^{-37}
$\frac{1}{5}$	6	50	2^{-33}	2^{-37}
$\frac{1}{5}$	6	100	2^{-65}	2^{-69}
$\frac{1}{10}$	20	50	2^{-64}	2^{-79}
$\frac{1}{10}$	10	50	2^{-71}	2^{-79}
$\frac{1}{10}$	10	100	2^{-145}	2^{-153}

For the opposite attack, $\mathcal{A}$ may intercept x_1 in Step 3 and then create a hidden chain w_2 that excludes x_1. With this attack strategy, $\mathcal{A}$ will broadcast x_1 to C_1 right before the timeout T_0 is reached, in hope that her hidden chain w_2 will outcompete the chain that the miners of C_1 begin to create. This attack vector is mitigated by disallowing a precomputation of w_2. Specifically, the enclave hardcodes b_1 into TX_2, and the predicate ϕ_2' verifies that b_1 is buried under at least ℓ_3 blocks.

The parameter ℓ_3 should be set to $2\ell_2 + T_1$. This gives a time span of T_1 blocks to update TX_1 into $\text{TX}_{1,1}$, after the enclave received the evidence that TX_1, TX_2 were confirmed and thus revealed x_1. The parameter T_1 should not be too low, to avoid the cancellation of the settlements in case of a short network outage or a slow chain growth in C_2 relative to C_1.

In the current Bitcoin network, $\ell_1 = 12$ suffices, hence the predicates ϕ_1', ϕ_2' require $\leq 12 + \ell_2 + \ell_3$ hash invocations for confidence level ℓ_2. Given that the complexity of ECDSA signature verification is an order of magnitude higher than that of invoking a hash function, moderate values such as $\ell_2 = 50$, $T_1 = 10$, $\ell_3 = 2\ell_2 + T_1 = 110$ imply that Bitcoin miners can validate the scripts ϕ_1', ϕ_2' for a mild fee. These parameters for PoW-based SPV proofs can be even better if the cryptocurrency system supports NIPoPoW [35, 65].

It is unlikely that Π_{theo} will be vulnerable to an attack that embeds a transaction that spends TX_1 into $\text{TX}_{1,1}$ or $\text{TX}_{1,2}$ in another cryptocurrency system C_3, where C_3 has the same PoW hash function and the same difficulty level. The reason is that the *txid* hash of TX_1 in the leaf of the Merkle tree is determined according to the prior history that goes back to the genesis block of C_1. Unless C_3 allows the input of a transaction to consist of arbitrary data, $\mathcal{A}$ will need to mount a preimage attack that creates valid transaction in C_3 with a particular value (i.e., the *txid* of TX_1) as its hash.

The main obstacle to an implementation of Π_{theo} in Bitcoin is the RMIT functionality. It is possible to implement the specific RMIT that Π_{theo} requires by creating a transaction tx_{init} that spends the inputs into a single output that is controlled by the secret signing key of Tesseract, and creating a refund transaction tx_{refund} that has `locktime` [6] of T_0 and spends the output of tx_{init} back into the inputs. After the enclave receives evidence that tx_{refund} is publicly available, it will broadcast tx_{init} to the Bitcoin network. When the execution of Π_{theo} reaches Step 3 and the enclave needs to release x_1, it will broadcast a transaction tx_{commit} that spends the output of tx_{init} into the desired outputs. The only problem with this procedure is that there is no good way to make tx_{refund} publicly available while relying on the security of Bitcoin alone. In a purely theoretical sense, it is possible to make tx_{refund} available by storing

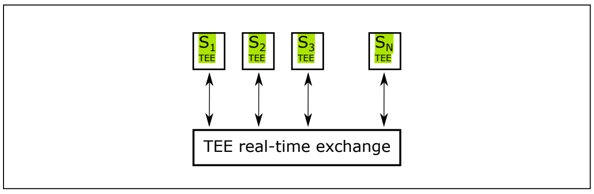


Figure 10: Practical fair settlement.

it as arbitrary data on the Bitcoin blockchain using OP_RETURN, but this will be very costly because the size of tx_{refund} can be dozens of kilobytes and the capacity of an OP_RETURN output is only 80 bytes. An efficient version of RMIT can be done via a Bitcoin protocol fork: an initial transaction will mark both the inputs and the new outputs as unspendable in the UTXO set, and a subsequent transaction will supply a witness to ϕ_1 or ϕ_2 and thereby ask the miners to make either the inputs or the outputs spendable (for a fee). An Ethereum implementation of a RMIT contract is possible, but it should be noted that Π_{theo} (and its generalization to more than two systems) requires RMIT support by all the cryptocurrency systems that are involved in the settlement.

Our analysis of Π_{theo} gives the essential security arguments for a protocol that enables an all-or-nothing settlement. A formal security proof of Π_{theo} (as well as Π_{prac} of Section 3.3) requires a rigorous model for the cryptocurrency consensus system — such as GKL [51] or PSS [86] — together with a rigorous model that is rich enough to express the scripting language that controls the users' coins (see, e.g., [78]). In Appendix C we provide a formal security proof (under certain assumptions) for the ACCSs protocol of Appendix B, that also serves to show several of the ingredients that a proof for Π_{theo} needs to incorporate.

3.3 Practical Protocol

The theoretical protocol Π_{theo} of Section 3.2 is resilient against an adversary who has total access to the server machine, except for the data that is inside the TEE-enabled CPU. Here, we present a practical protocol Π_{prac} for the all-or-nothing settlement problem that relaxes this resiliency aspect, but in fact offers better security in other respects.

Our strategy is to distribute the trust among N additional servers that are all running TEE enclaves (see Figure 10), and ensure that Π_{prac} satisfies Definition 3.2 if there exists at least one server $S_j \in \{S_1, S_2, \ldots, S_N\}$ that is beyond the reach of the adversary $\mathcal{A}$. That is to say, we assume that S_j can communicate with cryptocurrencies C_1,C_2 without interference.

The main idea of Π_{prac} is to emulate the essential characteristic of the theoretical protocol Π_{theo}, which is to wait for a proof that the settlement transaction TX_1 was either committed to C_1 or cancelled, and then do the same for the settlement transaction TX_2.

The settlement protocol Π_{prac} that Tesseract and the servers $S_1, S_2, \ldots, S_N$ execute is specified in Figure 11. As a prerequisite, the Tesseract server and $S_1, S_2, \ldots, S_N$ need to share a symmetric secret key K that is known only to their enclaves. The transactions $\text{TX}_1^{\text{c}}, \text{TX}_2^{\text{c}}$ are "cancellation" transactions that invalidate the settlement transactions TX_1, TX_2, respectively. In Bitcoin, TX_i^{c} can be implemented simply by spending one of the inputs of TX_i into a

取消tx，通过花销一个input

Protocol Π_{prac}

(1) Tesseract sends $ct = \text{encrypt}_K(\text{TX}_1, \text{TX}_2, \text{TX}_1^{\text{c}}, \text{TX}_2^{\text{c}})$ to $S_1, S_2, \ldots, S_N$.
(2) For every $i \in [N]$, Tesseract waits for acknowledgement from S_i that it received ct.
(3) Tesseract broadcasts TX_1 to C_1.
(4) Starting from the time at which it received ct in Step 1, each server $S_i \in \{S_1, S_2, \ldots, S_N\}$ inspects the next blocks of C_1
   - If S_i does not see TX_1 on C_1 within T_1 blocks, then it broadcasts TX_1^{c} to C_1.
   - If S_i sees that TX_1 has ℓ_2 extra confirmations on C_1, then it broadcasts TX_2 to C_2.
   - If S_i sees that TX_1^{c} has ℓ_2 extra confirmations on C_1, then it broadcasts TX_2^{c} to C_2.

Figure 11: Practical protocol for fair settlement.

new output that is identical to that input (this will cause $\text{TX}_i, \text{TX}_i^{\text{c}}$ to conflict with each other).

Thus, the protocol Π_{prac} seeks to preserve the property that TX_2 remains confidential inside the enclaves for as long as TX_1 is not yet confirmed. This property avoids the risk that $\text{TX}_i, \text{TX}_{3-i}^{\text{c}}$ will compete for confirmations at the same time, as that can easily violate the all-or-nothing requirement.

In the case that at least one server S_i is not under physical attack, we have that either TX_1 or TX_1^{c} will be broadcast to C_1 within T_1 blocks. As a consequence, either TX_1 or TX_1^{c} will be confirmed after $T_1 + \ell_2$ blocks. This allows S_i or one of the other non-adversarial servers to broadcast the appropriate transaction (i.e., TX_2 or TX_2^{c}) to the cryptocurrency system C_2, causing it to be confirmed too.

The adversary $\mathcal{A}$ may attempt to mount a race attack with a head start of T_1 blocks, by eclipsing one of the servers S_j. The attack can proceed as follows:

(1) $\mathcal{A}$ intercepts the data TX_1 that Tesseract reveals in Step 3 of Π_{prac}, and deactivates the Tesseract server.
(2) $\mathcal{A}$ eclipses the server S_j, and feeds it with a fake blockchain (generated by $\mathcal{A}$ herself) that contains TX_1.
(3) When the enclave of S_j becomes convinced that TX_1 was confirmed, it releases TX_2.
(4) $\mathcal{A}$ waits until TX_1^{c} is confirmed on C_1, and then broadcasts TX_2 to C_2.

As with Π_{theo}, the reason that $\mathcal{A}$ obtains a head start is that the honest participants wait for a duration of T_1 blocks before they attempt to invalidate TX_1, whereas $\mathcal{A}$ begins to create her fake chain immediately — see Section 3.2 and Table 2 for analysis. Note that the purpose of the cancellation transaction TX_2^{c} is to defeat this race attack, in the case that $\mathcal{A}$ fails to generate ℓ_2 blocks while the honest network generates $T_1 + \ell_2$ blocks.

In fact, it is more difficult for $\mathcal{A}$ to exploit the head start and attack Π_{prac}, than it is to attack Π_{theo}. This is because Π_{prac} can specify the precise duration T_1, and Π_{theo} has to estimate T_1 by setting T_0 in the predicate ϕ_2. This estimation should use a lenient bound (that will likely give $\mathcal{A}$ a larger head start), as otherwise the variance of the block generation process can cause ϕ_2 to be triggered and thus abort the settlement.

Notice that $\mathcal{A}$ cannot mount an eclipse attack before Step 3 of Π_{prac} is reached. Only the Tesseract enclave can produce the data

Table 3: Settlement transaction size and fee. In the columns of total cost and cost per user, the first value represents the cost on Bitcoin and the second one on Litecoin.

Number of active users	Size of settlement transaction (KB)	Total settlement cost (USD)	Settlement cost per user (USD)
10	2.538	10.004/0.192	1.000/0.019
100	23.674	93.312/1.788	0.933/0.018
1000	235.026	926.374/17.753	0.926/0.018
2000	469.887	1852.093/35.494	0.926/0.018
3000	704.766	2777.886/53.236	0.926/0.018
4000	939.640	3703.659/70.978	0.926/0.018

TX_1, and it will do so only after receiving all the acknowledgements from $S_1, S_2, \dots, S_N$ in Step 2. Therefore, an eclipse attack will be thwarted if at least one non-adversarial server $S_i \in \{S_1, S_2, \dots, S_N\}$ is present, because S_i will broadcast the invalidation transactions $\text{TX}_1^c, \text{TX}_2^c$ to ensure the all-or-nothing guarantee of Definition 3.1.

In practice, it is preferable that the Tesseract enclave will wait for acknowledgements from only a constant fraction of the servers $S_i \in \{S_1, S_2, \dots, S_N\}$, so that $\mathcal{A}$ will not be able to deny service by preventing a single acknowledgement from reaching Tesseract in Step 2 of the settlement procedure. Our practical approach can in fact make Tesseract resistant to DoS in a broader sense, via a consensus protocol among identical servers. Due to lack of space, we defer the full protocol to [27].

Another advantage of Π_{prac} is that it can support other cryptocurrency systems besides a PoW blockchain. This is because the servers $S_1, S_2, \dots, S_N$ can run a full node inside their enclave, whereas the predicates ϕ_1', ϕ_2' lack the power to express the irreversibility condition of a more complex cryptocurrency system (see Appendix D).

Irrespective of the settlement procedure, the Tesseract exchange server can fetch from $S_1, S_2, \dots, S_N$ the heights of their longest chains (e.g., once every 30 minutes), and refuse to confirm users' deposits if less than $N/2$ of the servers respond. This would avert fake deposits from being confirmed due to an eclipse attack, without relying on the prudence of the users.

4 IMPLEMENTATION AND EVALUATION

We implemented a prototype of Tesseract that executes the clients' trade orders, and performs all-or-nothing settlements. In this section, we present implementation details and evaluation results.

4.1 Real-time trading

We implemented a continuous limit-orderbook that runs fully protected inside the enclave. Figure 12 contains end-to-end measurements of the real-time trading performance of our prototype: For each level of concurrency, users concurrently send encrypted and signed orders to the enclave over the local network and we measure the throughput and latency over the course of processing 3 million orders. Each user repeatedly sends a randomly generated order, and then (synchronously) awaits a reply from the exchange (i.e. whether the order was (partially) filled and/or placed in the orderbook). The latency is the time between sending the order and receiving a response. The exchange is running on a recent model Intel CPU (i7-8700) using six threads, one per physical core.

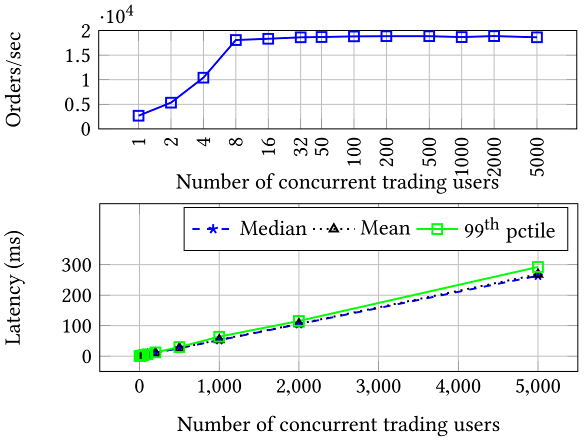


Figure 12: Trading performance

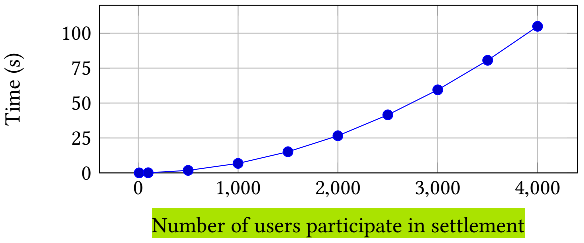


Figure 13: Time to generate a pair of settlement transactions.

As suggested by Figure 12, our prototype supports thousands of concurrent users, processing over 18k orders per second, with latency scaling linearly with the number of users; for 2k concurrent users we achieve 99th percentile latency of 106ms, for 5k concurrent users we achieve 99th percentile latency of 268ms. For low numbers of concurrent users, full throughput is not achieved due to idle exchange threads that are not being utilized by our synchronous benchmark.

4.2 All-or-nothing settlement

We implemented the atomic settlment protocol Π_{prac} between Bitcoin and Litecoin. To minimize the Trusted Computing Base (TCB), we ported only the necessary part of Bitcoin Core v0.14.0 to SGX, resulting in only using ~13.3% of it. We use NaCl [32] for lightweight secure channels, rather than TLS. The entire TCB of our implementation consists of approximately 850 source lines of code (SLoC) for the functionalities that run inside the SGX enclave. This figure excludes Bitcoin Core and NaCl code, which contribute about 10,284 SLoC and 1,057 SLoC, respectively.

The source code of our implementation demo is available at https://github.com/iddo333/exchSGX. We tested the correctness of our implementation by running it on the public testnets. The confirmed settlement transactions can be viewed at [10, 11].

Transaction size and fees. First, we evaluate the settlement transaction size and cost with respect to different number of active traders. Note that only users with trade activity (since the last settlement) take part in the settlement transactions. As shown in Table 3, the size of the settlement transactions grows linearly with the number of active user, while the transaction fee per user roughly remains constant. On average, it costs a single user approximately 92.6 cents to settle on Bitcoin and 1.8 cents on Litecoin. This results in a total daily cost of 94.4 cents for an active user trading between Bitcoin and Litecoin.

We estimate the transactions fees by calculating the average transaction fee per kilobyte from historic blockchain data. Specifically, we analyzed 10081580 transactions in blocks 587351 to 592040 (July 28, 2019 - Aug 27, 2019) using the Google BigQuery [42] Bitcoin dataset, and found the average unit cost of transaction fee on the Bitcoin network to be 0.00039 Bitcoins per kilobyte of transaction data (BTC/KB). Similarly, the average transaction fee on Litecoin is 0.00105 LTC/KB. The Bitcoin and Litecoin prices (on August 27th, 2019) were 10106.6 USD/BTC and 71.94 USD/LTC, respectively.

Transaction generation time. Figure 13 shows the total generation time of two settlement transactions (one for Bitcoin and one for Litecoin). The bottleneck in the generation of a settlement transaction is hashing a large amount of data, and signature computation. Our current implementation generates legacy transactions (i.e., before the recent SegWit [33] upgrade), hence the signing time is quadratic [2] in the number of inputs – each input requires re-hashing slightly different versions of the entire data, instead of only the signature computation. The structure of transactions on Bitcoin and Litecoin are the same, so the generation time of one settlement transaction inside the SGX enclave for either blockchain is approximately 6.6 seconds for 1000 users. Up to 5.7 seconds, 86.3% of the generation time, is spent on hashing.

Since all the trading is done off-chain and settlements occur periodically, the time for generating settlement transactions is minor, even when the number of active traders is large. E.g., for daily settlements with 1000 users that trade between 2 blockchains, transaction generation takes about 13.2 seconds once every 24 hours. Note that the size of a settlement transaction among 1000 active traders is 235 KB, i.e., about 23% of the Bitcoin block capacity (pre-SegWit), leaving 77% of the capacity for other commerce during the settlement timeframe. To compare, decentralized exchanges typically have only dozens of active traders per day on average [5, 8], since each trade is performed on-chain. Centralized exchanges can have 100k active traders per day [100].

With SegWit, the Bitcoin block capacity is doubled, and the transaction fees a typically 35% smaller [3, 111] as the quadratic hashing overhead is avoided. Thus, while our reference implementation may already be appealing to traders, it can be improved by migrating to SegWit (legacy code still predominates the Bitcoin ecosystem [9]). Transaction signing is also highly parallelizable, so with more engineering effort the generation time can be reduced by signing each input concurrently.

5 RELATED WORK

Trusted hardware has been proposed as an effective tool for different kinds of cryptocurrency use-cases, such as off-chain payment channels [69], reputable data feed services [114], and a mixing service [105]. These schemes offer better efficiency and features by placing more trust in the hardware manufacturer: in particular, off-chain channels and mixers can also be accomplished without secure processors (see, e.g., [28, 57, 74, 95]). By contrast, Tesseract reduces the amount of trust that needs to be placed in the exchange service relative to all other real-time exchange schemes (to the best of our knowledge). In Appendix A we provide a comparison between Tesseract and various other cryptocurrency exchange schemes.

Trusted hardware can also be used to achieve significant efficiency gains for well-known cryptographic primitives such as functional encryption [50], secure MPC [90], and NIZK in the presence of side-channels [104]. Pass, Shi, and Tramèr give a formal model of trusted hardware and remote attestation [87].

Several works achieve fair exchange and secure cash distribution via interaction with a cryptocurrency system, cf. [16, 17, 28, 66]. However, these works enable fair exchange (with penalties) by using a single cryptocurrency system, while Tesseract has to provide all-or-nothing fairness among multiple cryptocurrency systems.

Outside of academic work, a wide range of industry and community efforts have attempted to realize various aspects of cross-chain distributed exchange. Notable strategies include the use of payment channels to achieve a hub-and-spoke exchange, cf. Appendix A.

Several exchanges aim at using raw atomic swaps, as in Figures 15 and 16. Per Appendix B, such use of on-chain mediation is unsuitable for real-time trades. Further alternatives to an atomic swap model for decentralized exchanges are explored in Appendix A – e.g., the use of IOUs as the basis of an exchange platform.

A wide range of decentralized exchanges run inside a single blockchain and let users to swap assets on that chain, using custody in smart contracts to trustlessly hold user assets. Due to their on-chain settlement, these exchanges are not real-time, and suffer from several manipulation vectors across a design space explored in [40].

Acknowledgments. This work is funded in part by NSF grants CNS-1617676, CNS-1514163, CNS-1564102, and CNS-1704615, as well as ARO grant W911NF-16-1-0145.

REFERENCES

[1] [n.d.]. https://github.com/intel/linux-sgx/issues/161.
[2] [n.d.]. https://github.com/bitcoin/bips/blob/master/bip-0143.mediawiki.
[3] [n.d.]. https://www.buybitcoinworldwide.com/fee-calculator/.
[4] [n.d.]. https://github.com/ethereum/EIPs/blob/master/EIPS/eip-155.md.
[5] [n.d.]. 0x insights. https://chainactivity.io/0x-project/insights.
[6] [n.d.]. Bitcoin Glossary: Locktime. https://bitcoin.org/en/glossary/locktime.
[7] [n.d.]. BTC relay. http://btcrelay.org/.
[8] [n.d.]. Dapp Rankings. https://dappradar.com/rankings/protocol/ethereum/category/exchanges.
[9] [n.d.]. SegWit usage. https://blockchair.com/bitcoin/charts/segwit-usage?interval=full&granularity=week.
[10] 2019. Atomic settlement transactions (Bitcoin part). https://tbtc.bitaps.com/261ae86bd92c5805c8dfc36b4db4992a7595af90d991d766e0483a9e89ac08e9.
[11] 2019. Atomic settlement transactions (Litecoin part). https://tltc.bitaps.com/61915f3fa3de25cd67c210a9760ed2d06a28eaaa2d8775535b8d559bed3f4167.
[12] Hamza Abusalah, Joël Alwen, Bram Cohen, Danylo Khilko, Krzysztof Pietrzak, and Leonid Reyzin. 2017. Beyond Hellman's Time-Memory Trade-Offs with Applications to Proofs of Space. In *23rd ASIACRYPT*.
[13] Alexey Akhunov. [n.d.]. https://github.com/ledgerwatch/eth$_s$tate/.
[14] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. 2013. Innovative Technology for CPU Based Attestation and Sealing. In *HASP'13*. 1–7. https://doi.org/10.1.1.405.8266
[15] Gavin Andresen. [n.d.]. P2SH. https://github.com/bitcoin/bips/blob/master/bip-0016.mediawiki.

[16] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek. 2014. Fair Two-Party Computations via Bitcoin Deposits. In *FC*.

[17] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek. 2014. Secure Multiparty Computations on Bitcoin. In *IEEE S&P*.

[18] Sarah Azouvi, Patrick McCorry, and Sarah Meiklejohn. [n.d.]. Betting on Blockchain Consensus with Fantomette. https://arxiv.org/abs/1805.06786.

[19] Moshe Babaioff, Shahar Dobzinski, Sigal Oren, and Aviv Zohar. 2012. On Bitcoin and red balloons. In *ACM Conference on Electronic Commerce*. 56–73.

[20] Adam Back. 2013. $O(2^{80})$ theoretical attack on P2SH. https://bitcointalk.org/index.php?topic=323443.0.

[21] Clare Baldwin. [n.d.]. http://www.reuters.com/article/us-bitfinex-hacked-hongkong-idUSKCN10E0KP.

[22] Andrew Barisser. 2015. https://medium.com/on-banking/high-frequency-trading-on-the-coinbase-exchange-f804c80f507b.

[23] Massimo Bartoletti and Livio Pompianu. 2017. An analysis of Bitcoin OP_RETURN metadata. In *FC*. https://arxiv.org/abs/1702.01024.

[24] Jethro Beekman. 2014. A Denial of Service Attack against Fair Computations using Bitcoin Deposits. https://eprint.iacr.org/2014/911.

[25] Juan Benet. [n.d.]. https://ipfs.io/.

[26] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. 2016. Cryptocurrencies without Proof of Work. In *Financial Cryptography Bitcoin Workshop*.

[27] Iddo Bentov, Yan Ji, Fan Zhang, Lorenz Breidenbach, Philip Daian, and Ari Juels. 2017. Full Technical Report, Tesseract: Real-Time Cryptocurrency Exchange Using Trusted Hardware. https://eprint.iacr.org/2017/1153.

[28] Iddo Bentov, Ranjit Kumaresan, and Andrew Miller. [n.d.]. Instantaneous Decentralized Poker. In *Asiacrypt 2017*.

[29] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. [n.d.]. Proof of activity: extending Bitcoin's proof of work via proof of stake. In *NetEcon 2014*.

[30] Iddo Bentov, Alex Mizrahi, and Meni Rosenfeld. 2017. Decentralized Prediction Market without Arbiters. In *Financial Cryptography 4th Bitcoin Workshop*.

[31] Iddo Bentov, TierNolan, et al. 2013. Atomic transfers. https://bitcointalk.org/index.php?topic=193281.msg2224949#msg2224949.

[32] Daniel J. Bernstein, Tanja Lange, and Peter Schwabe. [n.d.]. The Security Impact of a New Cryptographic Library. In *LATINCRYPT 2012*.

[33] Bitcoin developers. 2019. Segregated Witness. https://en.bitcoin.it/wiki/Segregated$_{\mathrm{W}}$itness.

[34] Daniel G Brown. 2011. How I wasted too long finding a concentration inequality for sums of geometric variables. https://cs.uwaterloo.ca/browndg/negbin.pdf.

[35] Benedikt Bünz, Lucianna Kiffer, Loi Luu, and Mahdi Zamani. [n.d.]. Flyclient: Super-Light Clients for Cryptocurrencies. https://eprint.iacr.org/2019/226.

[36] CryptoAsset Market Capitalizations. [n.d.]. https://coinmarketcap.com/assets/.

[37] Clark, Bonneau, Felten, Kroll, Andrew Miller, and Narayanan. 2014. On Decentralizing Prediction Markets and Order Books. In *WEIS*.

[38] K. Croman, C. Decker, I. Eyal, A. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. Sirer, D. Song, and R. Wattenhofer. 2016. On Scaling Decentralized Blockchains. In *FC Bitcoin Workshop*.

[39] Leslie Culbertson. [n.d.]. https://newsroom.intel.com/editorials/protecting-our-customers-through-lifecycle-security-threats.

[40] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. [n.d.]. Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges.

[41] Phil Daian, Rafael Pass, and Elaine Shi. 2019. Snow White: Robustly Reconfigurable Consensus and Applications to Provably Secure Proofs of Stake. *FC*.

[42] Allen Day and Colin Bookman. 2018. Bitcoin in BigQuery: blockchain analytics on public data. https://cloud.google.com/blog/products/gcp/bitcoin-in-bigquery-blockchain-analytics-on-public-data.

[43] Christian Decker and Roger Wattenhofer. 2015. A Fast and Scalable Payment Network with Bitcoin Duplex Micropayment Channels. In *17th SSS*.

[44] Desmedt and Frankel. 1989. Threshold Cryptosystems. In *CRYPTO*.

[45] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. 2004. Tor: The Second-Generation Onion Router. In *13th Usenix Security*.

[46] dree12 (pseudonym). [n.d.]. List of Major Bitcoin Heists, Thefts, Hacks, Scams, and Losses. https://bitcointalk.org/index.php?topic=576337.

[47] Devdatt P. Dubhashi and Alessandro Panconesi. 2009. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge Uni. Press.

[48] Tuyet Duong, Lei Fan, Thomas Veale, and Hong-Sheng Zhou. [n.d.]. Securing Bitcoin-like Backbone Protocols against a Malicious Majority of Computing Power. 2016 ([n. d.]). http://eprint.iacr.org/2016/716

[49] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. 2015. Proofs of Space. In *CRYPTO*.

[50] Ben A. Fisch, Dhinakaran Vinayagamurthy, Dan Boneh, and Sergey Gorbunov. 2017. Iron: Functional Encryption using Intel SGX.

[51] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. 2015. The Bitcoin Backbone Protocol: Analysis and Applications. In *Eurocrypt*.

[52] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. 2016. Threshold-Optimal DSA/ECDSA Signatures. In *14th ACNS*.

[53] Arthur Gervais and Rami Khalil. 2018. The Liquidity Network. https://liquidity.network/whitepaper$_{\mathrm{L}}$iquidity$_{\mathrm{N}}$etwork.pdf.

[54] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In *26th Symposium on Operating Systems Principles*.

[55] Sharon Goldberg, Ethan Heilman, and other. 2018. Arwen. https://www.arwen.io/.

[56] BitFury Group. 2015. http://bitfury.com/content/5-white-papers-research/pos-vs-pow-1.0.2.pdf.

[57] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. 2017. TumbleBit. In *NDSS*. https://eprint.iacr.org/2016/575.

[58] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. 2015. Eclipse Attacks on Bitcoin's Peer-to-Peer Network. In *24th Usenix Security*.

[59] Maurice Herlihy. 2018. Atomic Cross-Chain Swaps. In *PODC*.

[60] Matthew Hoekstra, Reshma Lal, Pradeep Pappachan, Vinay Phegade, and Juan Del Cuvillo. 2013. Hasp, http://dl.acm.org/citation.cfm?doid=2487726.2488370.

[61] SP Johnson, VR Scarlata, C Rozas, E Brickell, and F Mckeen. 2016. https://software.intel.com/en-us/blogs/2016/03/09/intel-sgx-epid-provisioning-and-attestation-services.

[62] Keystone. [n.d.]. https://keystone-enclave.org/.

[63] Rami Khalil, Arthur Gervais, and Guillaume Felley. [n.d.]. TEX - A Securely Scalable Trustless Exchange. https://eprint.iacr.org/2019/265.

[64] Aggelos Kiayias, Ioannis Konstantinou, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In *CRYPTO*.

[65] Aggelos Kiayias, Andrew Miller, and Dionysis Zindros. 2017. Non-interactive proofs of proof-of-work. https://eprint.iacr.org/2017/963.

[66] Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas. 2015. Fair and Robust Multi-Party Computation using a Global Transaction Ledger. In *Eurocrypt*.

[67] Sophie Knight. [n.d.]. http://www.reuters.com/article/us-bitcoin-mtgox-wallet-idUSBREA2K05N20140321.

[68] Johnson Lau. [n.d.]. https://github.com/jl2012/bips/blob/vault/bip-0VVV.mediawiki.

[69] Joshua Lind, Ittay Eyal, Florian Kelbert, Oded Naor, Peter R. Pietzuch, and Emin Gün Sirer. 2018. Teechain. In *11th SYSTOR*.

[70] Loi Luu and Yaron Velner. 2017. KyberNetwork White Paper. https://kyber.network/assets/KyberNetworkWhitepaper.pdf.

[71] mappum (pseudonym). 2015. Mercury – Fully trustless cryptocurrency exchange. https://bitcointalk.org/index.php?topic=946174.0.

[72] Sinisa Matetic, Mansoor Ahmed, Kari Kostiainen, Aritra Dhar, David Sommer, Arthur Gervais, Ari Juels, and Srdjan Capkun. 2017. ROTE. http://eprint.iacr.org/2017/048.

[73] McCorry, Heilman, and Miller. [n.d.]. Atomically Trading with Roger: Gambling on the success of a hardfork. http://eprint.iacr.org/2017/694.

[74] Patrick McCorry, Malte Möser, Siamak Fayyaz Shahandashti, and Feng Hao. 2016. Towards Bitcoin Payment Networks. In *ACISP*.

[75] McKeen, Alexandrovich, Berenzon, Rozas, Shafi, Shanbhogue, and Savagaonkar. 2013. Innovative instructions and software model for isolated execution. In *HASP*.

[76] Robert McMillan. 2013. $1.2M Hack Shows Why You Should Never Store Bitcoins on the Internet. https://www.wired.com/2013/11/inputs/.

[77] Danielle Meegan. [n.d.]. https://www.ethnews.com/relay-attack-leads-to-etc-loss-on-ethereum-exchange.

[78] Andrew Miller. 2016. *Provable Security for Cryptocurrencies*. Ph.D. Dissertation. University of Maryland, College Park.

[79] Tal Moran and Ilan Orlov. 2019. Rational Proofs of Space-Time. *Crypto* (2019).

[80] Sebastian Muller, Franziska Brecht, Benjamin Fabian, Steffen Kunz, and Dominik Kunze. 2012. Distributed performance measurement and usability assessment of the tor anonymization network. In *Future Internet*, Vol. 4(2). 488–513.

[81] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).

[82] Satoshi Nakamoto. 2010. https://bitcointalk.org/index.php?topic=1786.msg22119#msg22119.

[83] Chia Network. 2018. https://chia.network/.

[84] NIST. 2018. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf.

[85] Sunoo Park, Krzysztof Pietrzak, Albert Kwon, Joël Alwen, Georg Fuchsbauer, and Peter Gazi. 2015. Spacemint: A Cryptocurrency Based on Proofs of Space. *IACR Cryptology ePrint Archive* 2015 (2015), 528. http://eprint.iacr.org/2015/528

[86] Rafael Pass, Lior Seeman, and abhi shelat. 2017. Analysis of the Blockchain Protocol in Asynchronous Networks. In *Eurocrypt*.

[87] Rafael Pass, Elaine Shi, and Florian Tramer. 2017. Formal Abstractions for Attested Execution Secure Processors. In *Eurocrypt*.

[88] Andrew Poelstra, Adam Back, Mark Friedenbach, Gregory Maxwell, and Pieter Wuille. 2017. Confidential Assets. In *FC Bitcoin Workshop*.

[89] Poon and Dryja. [n.d.]. https://lightning.network/lightning-network-paper.pdf.

[90] Portela, Barbosa, Scerri, Warinschi, Bahmani, Brasser, and Sadeghi. 2017. Secure Multiparty Computation from SGX. In *FC*.

[91] Portnoy and Eckersley. [n.d.]. https://www.eff.org/deeplinks/2017/05/intels-management-engine-security-hazard-and-users-need-way-disable-it.
[92] profitgenerator. 2017. EtherDelta. https://steemit.com/ethereum/@profitgenerator/etherdelta-decentralized-token-exchange.
[93] Meni Rosenfeld. 2012. Colored Coins. https://bitcoil.co.il/files/Colored%20Coins.pdf and https://bitcoil.co.il/BitcoinX.pdf.
[94] Meni Rosenfeld. 2014. http://arxiv.org/abs/1402.2009.
[95] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. 2017. P2P Mixing and Unlinkable Bitcoin Transactions. In *NDSS 2017*.
[96] Fabian Schuh and Daniel Larimer. [n.d.]. BitShares. https://bravenewcoin.com/assets/Whitepapers/bitshares-financial-platform.pdf.
[97] Felix Schuster, Manuel Costa, Cédric Fournet, Christos Gkantsidis, Marcus Peinado, Gloria Mainar-Ruiz, and Mark Russinovich. 2015. VC3. In *IEEE S&P*.
[98] ShapeShift. [n.d.]. https://shapeshift.io/.
[99] Spacemesh. [n.d.]. https://spacemesh.io/.
[100] Tony Spilotro. 2018. Only 4 Crypto Exchanges Have 100,000+ Active Users. https://www.newsbtc.com/2018/12/12/crypto-exchanges-active-users/.
[101] Raoul Strackx and Frank Piessens. 2016. Ariadne: A Minimal Approach to State Continuity. In *25th USENIX Security*.
[102] Paul Sztorc. 2015. http://www.truthcoin.info/blog/bitusd/.
[103] Todd. [n.d.]. https://github.com/bitcoin/bips/blob/master/bip-0065.mediawiki.
[104] Florian Tramer, Fan Zhang, Huang Lin, Jean-Pierre Hubaux, Ari Juels, and Elaine Shi. 2017. Sealed-Glass Proofs. In *Euro S&P*.
[105] Muoi Tran, Loi Luu, Min Suk Kang, Iddo Bentov, and Prateek Saxena. 2018. Obscuro: A Secure and Anonymous Bitcoin Mixer using SGX. In *ACSAC*.
[106] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. 2018. Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-order Execution. In *USENIX*.
[107] Warren and Bandeali. [n.d.]. https://0xproject.com/pdfs/0x$_w$hite$_p$aper.pdf.
[108] Pieter Wuille et al. [n.d.]. https://bitcoincore.org/en/2017/03/23/schnorr-signature-aggregation/.
[109] Pieter Wuille, Gregory Maxwell, et al. [n.d.]. https://github.com/bitcoin-core/secp256k1.
[110] Xu, Cui, and Peinado. 2015. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *IEEE S&P*.
[111] Joseph Young. [n.d.]. https://www.newsbtc.com/2017/11/10/54991/.
[112] Joseph Young. 2016. https://cointelegraph.com/news/china-imposes-new-capital-controls-bitcoin-price-optimistic.
[113] Alexei Zamyatin, Dominik Harz, Joshua Lind, Panayiotis Panayiotou, Arthur Gervais, and William J. Knottenbelt. [n.d.]. XCLAIM: Trustless, Interoperable Cryptocurrency-Backed Assets. https://eprint.iacr.org/2018/643.
[114] Fan Zhang, Ethan Cecchetti, Kyle Croman, Ari Juels, and Elaine Shi. 2016. Town Crier: An Authenticated Data Feed for Smart Contracts. In *CCS*.
[115] Fengwei Zhang and Hongwei Zhang. 2016. SoK: A Study of Using Hardware-assisted Isolated Execution Environments for Security *(HASP)*.
[116] ZIP143. [n.d.]. https://github.com/zcash/zips/blob/master/zip-0143.rst.

A CRYPTOCURRENCY EXCHANGES

We describe several alternative designs for a real-time cryptocurrency exchange, and also survey non-real-time designs. See Table 4 for a summary comparison.

A.1 Centralized Exchange

In a centralized cryptocurrency exchange, users transfer ownership of their funds to the sole control of the exchange administrator. This transfer of ownership (a.k.a. deposit) is done via an on-chain transaction that may take a long time to be confirmed, according to a confidence parameter that the exchange administrator set. Most exchanges accept a Bitcoin transfer by waiting 1 hour on average (6 PoW confirmations).

The business model of a centralized exchange can be described as a "goose that lays golden eggs". That is to say, the exchange administrator may run away with all the funds that the users deposited (usually by claiming "I was hacked"), and the disincentive to doing so is that the exchange collects a fee from each trade between the users. Most exchanges also charge a withdrawal fee, and some exchanges collect fees even when the users place bid and ask orders.

Still, there have been many thefts of funds from centralized exchanges (cf. [46]). About 650,000 bitcoins were lost when the MtGox exchange shut down in 2014 [67], and the users of the Bitfinex exchange lost approximately 120,000 bitcoins in 2016 [21].

A.2 Exchange Based on Multisig with TTP

An exchange design under which the traders' funds cannot be stolen is described in [30, Appendix A]. The idea is that each trader will deposit her assets into a script that is controlled both by her and by a semi trusted third party (TTP). Traders will then communicate their trades in real-time to the TTP, and the TTP is supposed to keep honest accounting off-chain. Periodically, the traders and the TTP will cooperate to sign the new state after all the trades that have been made, and broadcast the result to the blockchain.

This process is highly susceptible to DoS by malicious traders who would abort instead of signing the new state. Therefore, the exchange may require splitting the traders into smaller factions, or impose penalties on misbehaving traders who refuse to sign a new state. However, such penalties would require additional collateral from traders who wish to trade with a relatively modest amount of funds (since a malicious trader can perform an abort attack by sacrificing her funds), which makes the exchange service less attractive.

As with any TTP-based scheme, this design is susceptible to frontrunning attacks by a dishonest TTP (cf. Section 2.4).

A.3 Exchange via Off-chain Channels and TTP

In this design, each user establishes off-chain bi-directional payment channels [43, 74, 89] with a semi-TTP server S, one channel for each cryptocurrency that the user wishes to trade in. This produces a hub and spoke network structure, see Figure 14 for an illustration of BTC/LTC/ETH trading.

The traders will then communicate their bid and ask orders to S. Whenever the orders of two traders match, they will send an instant off-chain payment to S, and S will route the funds of one trader to the other.

It is better for each individual to trade in small amounts, because the TTP can always steal the most recent amount that was funneled through S. However, this recommendation is in conflict with the common behavior of large traders, who frequently create big bid/ask "walls".

In any case, even if the amount in each trade is small, the risk of theft by a corrupt TTP remains high. This is because the aggregate amount that all the traders funnel through S at a particular point in time can be substantial. An example that does not involve an exchange but demonstrates this point is the online wallet service inputs.io, which made it attractive for users to deposit small amounts and then ran away with more than 4000 bitcoins [76].

Another major drawback of this approach is that the TTP has to lock matching collateral for each off-chain payment channel of each trader, due to nature of off-chain bi-directional channels. It is therefore likely that the exchange service would need to impose high fees on its users.

The Liquidity Network [53] and Arwen [55] are exchange platforms that are based on off-chain channels with TTPs.

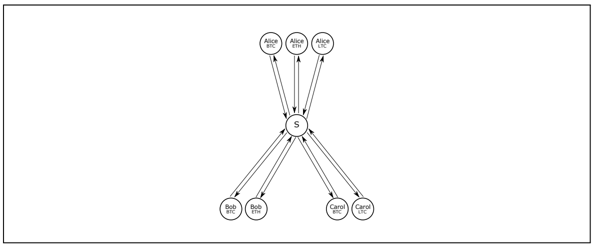


Figure 14: Exchange via off-chain channels.

Table 4: Comparison of Cryptocurrency Exchanges

	Trust	DoS	Collateral	Front-running	Price Discovery
Centralized	yes	minor	no	yes	yes
TTP/multisig (Appendix A.2)	minor	yes	from users	yes	yes
TTP/channels (Appendix A.3)	semi	minor	from TTP	yes	yes
ShapeShift	semi	minor	no	yes	no
Tesseract	TEE	minor	no	no	yes

A.4 Non-real-time Exchanges

ShapeShift [98] is a centralized matching service that mitigates the risks associated with a full-fledged exchange by necessitating that each trader will deposit only a small amount of cryptocurrency for a short period of time. If a quick match is available then ShapeShift will execute the trade, otherwise it will immediately refund the cryptocurrency to the trader (i.e., via a transaction on the blockchain). ShapeShift does not support real-time trades and price discovery. It fetches the current prices from centralized exchanges. Since ShapeShift is rather popular, the aggregated amount of funds that can be stolen is likely to be substantial. In this sense, ShapeShift does not solve the systemic risk that centralized exchanges entail.

EtherDelta [92] is a non-real-time non-cross-chain decentralized exchange that has been operational since 2016, with quite a significant amount of popularity—particularly for the first listings of Initial Coin Offerings (ICO). However, EtherDelta is vulnerable to frontrunning attacks, see [40].

BitShares [96] offers a cryptocurrency exchange that does not rely on trusted parties. It is not real-time, but relatively fast due to a delegated proof-of-stake consensus protocol in which blocks are created every few seconds by central committee members (who may engage in frontrunning attacks, see Section 2.4). Traders first convert their cryptocurrency to IOUs in the BitShares system, and later convert these IOUs to the native BitShares cryptocurrency (BTS) according to an up-to-date exchange rate that is set by elected representatives that the BitShares stakeholders voted for. See [96, Section 2] and [102] regarding the risk of market manipulation with this approach. The BTS cryptocurrency that traders ultimately obtain can be exchanged for other cryptocurrencies by means that are again external to the BitShares system — centralized exchanges (a.k.a. gateways) are commonly used for this task.

Exchange Based on Mutual Distrust: Instead of relying on trusted hardware, it would be possible in principle to operate an exchange service (similar to Tesseract) as a logical server that is implemented via multiple physical servers that are distrustful of each other. Traders will need to send their bid/ask requests using threshold encryption [44] in order to avoid frontrunning attacks (see Section 2.4), and the physical servers will run a Byzantine consensus protocol and sign the settlement transactions (cf. Section 2) with a threshold signature scheme [52]. An honest majority among the physical servers can guarantee protection from theft. Since the physical servers would need to reside in different geographical locations to provide meaningful security, and since Byzantine agreement with threshold decryption has to be performed for each of the users' orders, the latency of a mutual distrust based exchange would probably be measured in seconds (depending on the number of physical servers). By contrast, the responsiveness of Tesseract can be measured in milliseconds.

B ATOMIC CROSS-CHAIN SWAPS

A secure protocol for ACCSs was given in [31]. We specify an intuitive description of the protocol in Figure 15, demonstrating a swap of bitcoins for litecoins as an example. The main thrust of the protocol Π_{accs} is that Alice can redeem Bob's coins only by publicly revealing her decommitment x on a blockchain, thereby allowing Bob to use x to redeem Alice's coins on the other blockchain. To avoid a race condition, Alice's coins remain locked for s_0 more time than Bob's coins, which should give Bob enough time to learn x and claim Alice's coins. The reason behind the time limits is that an honest party should be able to gain back possession of her money in the case that the other party aborted. We provide a proof of security for Π_{accs} in Appendix C.

The first two steps of Π_{accs} terminate after c_0 and $f(c_0)$ confirmations on the Bitcoin and Litecoin blockchains, so that the transactions will become irreversible with a high enough probability. The function $f(\cdot)$ estimates a level of confidence for TX_B's irreversibility that is on par with that of TX_A. Per Appendix A.1, a reasonable choice for $f(\cdot)$ can be, e.g., $f(n) = 3n$. Combined with a sensible choice for the parameters t_0, s_0 (see Appendix C), Alice and Bob will need to wait for hours (or perhaps minutes with faster cryptocurrency systems) until the Π_{accs} protocol completes.

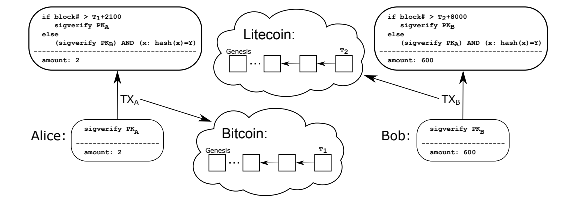
In the accompanying illustration (Figure 16), Alice trades $n_1 = 2$ BTC for Bob's $n_2 = 600$ LTC. The last block of the Bitcoin blockchain is T_1, and the last block of the Litecoin blockchain is T_2. The time limit t_0 is set to about two weeks into the future (i.e., 2000 more blocks in Bitcoin, and 8000 more blocks in Litecoin, as the block creation rate is 4 times faster in Litecoin than in Bitcoin). The extra safety time s_0 is set to 100 Bitcoin blocks, which is ≈ 16 hours on average. Note that both Bitcoin and Litecoin allow specification of the time limit in seconds rather than blocks (since valid blocks need to specify a timestamp that is within certain leniency bounds), which adds convenience but not security.

Since the long confirmation time in decentralized networks makes Π_{accs} slow, it is likely that the agreed upon price (in the example, $n_2/n_1 = 300$ LTC per BTC) was decided by observing the prices in real-time exchanges. This implies that the parties cannot respond to price fluctuations in a fair manner: if Bob is rational then he may cancel the trade after the first step (if the market price of LTC went up), and if Alice is rational then she may cancel the trade

Protocol Π_{accs}

1. Alice samples a random $x \in \{0,1\}^\lambda$, computes a hash commitment $Y = \text{hash}(x)$, and broadcasts a transaction TX_A that spends n_1 BTC into an output script that dictates:
   - Alice can gain back possession of her n_1 BTC after $c_0 + t_0 + s_0$ blocks.
   - Bob can redeem the n_1 BTC by supplying a preimage of Y and signing with his secret key.
2. After TX_A is buried under c_0 extra blocks and therefore becomes irreversible w.h.p., Bob broadcasts a transaction TX_B that spends his n_2 LTC into an output script that dictates:
   - Bob can gain back possession of his n_2 LTC after $4t_0$ blocks.
   - Alice can redeem the n_2 LTC by supplying a preimage of Y and signing with her secret key.
3. After TX_B is buried under $f(c_0)$ extra blocks and therefore becomes irreversible w.h.p., Alice redeems the n_2 LTC of Bob by revealing x.
4. Bob supplies x to redeem the n_1 BTC of Alice.

Figure 15: Protocol for an atomic cross-chain swap.



```
if block# > T1+2100
   sigverify PKA
else
   (sigverify PKB) AND (x: hash(x)=Y)
--------------------------------------
amount: 2
```

```
if block# > T2+8000
   sigverify PKB
else
   (sigverify PKA) AND (x: hash(x)=Y)
--------------------------------------
amount: 600
```

```
sigverify PKA
--------------------
amount: 2
```

```
sigverify PKB
--------------------
amount: 600
```

Figure 16: Illustration of an atomic cross-chain swap.

after the second step (if the market price of BTC went up). Another implication is that Π_{accs} by itself is not a complete trading solution, because real-time exchanges are still needed for price discovery.

A matching service for ACCSs was established in 2015, though it became defunct due to lack of usage [71].

C PROOF OF SECURITY FOR ACCS

Per Definition 3.1, let us prove that the all-or-nothing requirement holds for the Π_{accs} protocol of Appendix B.

We use $\text{TXOUT}_\text{A}, \text{TXOUT}_\text{B}$ to denote the outputs of the transactions $\text{TX}_\text{A}, \text{TX}_\text{B}$, respectively. We denote by $\text{TX}_\text{A}^\text{S}, \text{TX}_\text{B}^\text{S}$ the transactions that spend TXOUT_A and TXOUT_B in steps 3 and 4 of Π_{accs}, respectively.

Proposition C.1. *Assume that $s_0 = \Omega(\sqrt{t_0})$, and that any Bitcoin client that wishes to submit a valid transaction will be able to broadcast the transaction and have it included in one of the next s_0 blocks. Assume that the probability of reversing c_0 Bitcoin blocks or $4c_0$ Litecoin blocks is negligible. Let E_0 denote the event that the all-or-nothing property holds w.r.t. the transactions TX_A^S and TX_B^S. If* hash(·) *is preimage-resistant and the signature scheme is existentially unforgeable, then $\neg E_0$ occurs with negligible probability.*

Proof sketch. We define the following events:

- $E_1 = \{\text{TX}_\text{A}$ was reversed after Bob broadcasted $\text{TX}_\text{B}\}$
- $E_2 = \{\text{TX}_\text{B}$ was reversed after Alice revealed $x\}$
- $E_3 = \{$Bob spent TXOUT_A before Alice revealed $x\}$
- $E_4 = \{$Alice spent both TXOUT_A and TXOUT_B without forging a signature$\}$
- $E_F = \{$The adversary forged a signature$\}$
- $E_A = \{\text{TX}_\text{A}^\text{S}$ was confirmed by the Bitcoin network$\}$
- $E_B = \{\text{TX}_\text{B}^\text{S}$ was confirmed by the Litecoin network$\}$

It is enough to prove that $\Pr[\neg E_0 \cap \neg E_F]$ is negligible, because $\Pr[E_F]$ is negligible by assumption and

$$\begin{aligned} \Pr[\neg E_0] &= \Pr[(\neg E_0 \cap E_F) \cup (\neg E_0 \cap \neg E_F)] \\ &\le \Pr[E_F] + \Pr[\neg E_0 \cap \neg E_F] \,. \end{aligned}$$

Assume that E_F did not occur. If Alice redeems TXOUT_B then Bob will be able to redeem TXOUT_A unless either the block that contains TX_A was reversed on the Bitcoin blockchain (event E_1), or TXOUT_A was spent after the $c_0 + t_0 + s_0$ timeout expired (event E_4). More formally, we have $E_A \cap \neg E_B \cap \neg E_F \subseteq E_1 \cup E_4$.

Assume again that E_F did not occur. If Bob redeems TXOUT_A then Alice will be able to redeem TXOUT_B unless either the block that contains TX_B was reversed on the Litecoin blockchain (event E_2), or TXOUT_B never appeared on the Litecoin blockchain (event E_3). More formally, we have $E_B \cap \neg E_A \cap \neg E_F \subseteq E_2 \cup E_3$.

Therefore, we obtain

$$\begin{aligned} & \Pr[\neg E_0 \cap \neg E_F] \\ = \; & \Pr\big[\big((E_A \cap \neg E_B) \cup (E_B \cap \neg E_A)\big) \cap \neg E_F\big] \\ \le \; & \Pr[E_A \cap \neg E_B \cap \neg E_F] + \Pr[E_B \cap \neg E_A \cap \neg E_F] \\ \le \; & \Pr[E_1 \cup E_4] + \Pr[E_2 \cup E_3] \\ \le \; & \Pr[E_1] + \Pr[E_2] + \Pr[E_3] + \Pr[E_4] \,. \end{aligned}$$

By assumption, $\Pr[E_1]$ and $\Pr[E_2]$ are negligible since c_0 is large enough. Furthermore, $\Pr[E_3] = \text{negl}(\lambda)$ because the event E_3 implies that Bob computed a preimage of $\text{hash}(Y)$.

To bound $\Pr[E_4]$, we need to consider the event that the Bitcoin chain grew by $t_0 + s_0$ blocks before the Litecoin chain grew by $4t_0$ blocks. If this event occurs, then Alice will be able to redeem TXOUT_A first, and still have enough time to redeem TXOUT_B too. Note that the Bitcoin network is expected to generate only t_0 blocks by the time that the Litecoin network generated $4t_0$ blocks.

Let $Z = Z(t_0 + s_0, \frac{1}{5})$ be a random variable with negative binomial distribution that counts the total number of blocks that both the Bitcoin and Litecoin networks generated by the time that the Bitcoin network generated $t_0 + s_0$ blocks, hence $\text{E}[Z] = 5(t_0 + s_0)$. By using a standard tail inequality [34, 47] for the *binomial* distribution $B(\mu \cdot \text{E}[Z], \frac{1}{5})$ with $\mu \triangleq \frac{t_0}{t_0 + s_0}$, we obtain

$$\begin{aligned} \Pr[E_4] &= \Pr[Z < 5t_0] = \Pr[Z < \mu \cdot \text{E}[Z]] \\ &= \Pr\left[B(\mu \cdot \text{E}[Z], \frac{1}{5}) > t_0 + s_0\right] \\ &< e^{-\frac{1}{3}(\frac{1}{\mu}-1)^2 \mu (t_0 + s_0)} \\ &= e^{-\frac{1}{3}(s_0/t_0)^2 \cdot t_0} = e^{-\frac{1}{3}{s_0}^2/t_0} \,. \end{aligned}$$

Thus, $s_0 = \lambda\sqrt{t_0}$ implies $\Pr[E_4] < e^{-\lambda^2/3} = \text{negl}(\lambda)$. □

Proposition C.1 makes the assumption that clients cannot be denied from communicating with the Bitcoin network during a long enough time period. While DoS attack on clients has been suggested as a possible vulnerability of Bitcoin based protocols [24], our assumption is quite reasonable as it is far more difficult to mount a DoS attack on a client (that can connect to the internet from various endpoints) in comparison to a DoS attack on a server. However, in case the Bitcoin blocks approach their full capacity due to a high transaction volume, the client may indeed find it difficult to incorporate the desired transaction in one of the next s_0 blocks (see for example [38] regarding the scalability prospects of Bitcoin). Still, the client should be able to include her transaction by attaching a high enough fee and thus signal the Bitcoin miners to prioritize the transaction.

Notice that the chain growth ratio between Litecoin and Bitcoin (i.e., the constant 4) does not influence the proof, because the extra s_0 confirmations in $\mathrm{TXOUT_A}$ correspond to $4s_0$ expected growth that $\mathrm{TXOUT_B}$ precludes.

Let us also note that the above proof makes the implicit supposition that the computational power that is devoted to the Bitcoin and Litecoin networks remains constant. It is possible to generalize Proposition C.1 by assuming that the computational power may not fluctuate beyond a certain bound.

D DESIGN DETAILS

In additional to the high-level design description of Section 2, let us provide more particular details here.

For each supported cryptocurrency, the size of the FIFO queue of block headers is chosen according to a trade-off between complexity and security. For instance, 8064 Bitcoin block headers would correspond to a 2-month window (when header 8065 is added the first header will be removed, and so on), which means that forks that represent a period of time longer than 2 months cannot be supported. We note that Bitcoin and Litecoin block headers are 80 bytes each, and an Ethereum block header is ≤ 512 bytes.

In cryptocurrencies such as Bitcoin and Litecoin, the time limit of the deposit transactions (such as $\mathrm{TX_A}$ of Figure 1) can be expressed in the output script via the `CHECKLOCKTIMEVERIFY` instruction [103]. Technically, $\mathrm{SK_{TEEBTC}}$ can still spend the output after the time limit (since Bitcoin transactions should be *reorg safe* [82, 103]), but this is not guaranteed because the user may also spend the output then.

With regard to the random deposit addresses that the enclave generates, it is in fact better that a deposit address is a hash of the public key, as this increases security and reduces the size of unspent outputs on the public ledger. For example, a 257-bit compressed ECDSA public key gives 128 bits of security at most, while 160-bit hash digest of the 257-bit public key will give 160 bits of security (if the hash function is preimage-resistant). This is done in our implementation via P2SH [15] (P2WPK/P2WSH [68] can be used post-SegWit). Note that there is no point in mounting a collision attack on a scriptless address [20]. The settlement transaction will expose the public key, but potential attacks would then have a short timeframe until the transaction becomes irreversible. Hence, for maximal security the enclave will generate and attest to a fresh deposit address after each settlement.

Upcoming Bitcoin support for aggregated Schnorr signatures [108] will enable Tesseract to attach a single signature to the settlement transaction, instead of one signature for every input. This implies that the settlement transaction size can be halved, which is significant for large transactions (e.g., with 1000 traders the transaction size will 64 kilobytes smaller). It is also likely that miners will impose a considerably lower fee for a large settlement transaction with a single aggregated signature. At the level of principle, signature aggregation is beneficial since our secure enclave design requires Tesseract to refresh its deposit address (hashed public key) after each settlement, and hence the aggregated signature will need to be verified against different public keys. This is in contrast to a simpler scheme in which the enclave would have a single deposit address for all users at all times, where one ordinary signature for the entire settlement transaction would be enough (if the underlying cryptocurrency had built-in support for spending multiple inputs with the same signature).

In case of a forthcoming hardfork of the kind that created Ethereum Classic or Bitcoin Cash, users can secure themselves against replay attacks (cf. [73, Section 2.4] and [77]) by withdrawing their coins from the Tesseract exchange. Specifically, a malicious operator of the Tesseract server can feed the enclave blocks from the less valuable hardfork, then deposit and withdraw the less valuable coins, and then replays the withdrawal transaction to obtain coins from the account of the more valuable hardfork. For Tesseract, this attack can succeed in the account model but not in the UTXO model (as opposed to centralized exchanges where the attack may succeed in the UTXO model too), because the settlement transaction will have the new deposit in the inputs. The mechanism of Section 2.3 prevents this attack if the less valuable hardfork has much less hashpower (which is presumably the case for a hardfork with little value). Still, proper protection against hardfork replays can be supported if the hardforks themselves have built-in `chain_id` protection. Since hardfork replay attacks are well known at this point, many cryptocurrencies already implement the protection (Ethereum [4], ZCash [116], etc.). Our enclave's light clients handle this protection upon parsing the relevant deposit transactions (no need to parse data of the entire block). The users may switch to a new version of Tesseract with updated code that supports the hardfork (or completely new cryptocurrencies), and which can be deployed at a later time. Our reference demo has a preliminary Ethereum contract with dynamic support for ERC20 tokens, hence no switch is needed for new ERC20 tokens (users can create new order book pairs, for a fee).

Permissionless cryptocurrencies can be based on scarce resources other than PoW. In particular, in a proof-of-stake [18, 26, 29, 41, 48, 54, 56, 64] based cryptocurrency the scarce resources are the coins that circulate in the system, and in a proof-of-space [12, 49, 79, 83, 85, 99] based cryptocurrency the scarce resource is storage space. While our reference implementation of Tesseract currently supports only PoW based cryptocurrencies, we note that blockchain based proof-of-stake cryptocurrencies can be supported in a similar manner. Typically, the blocks of a PoW blockchain are validated by inspecting a hash digest, and blocks in a proof-of-stake blockchain are validated by inspecting the UTXO set (i.e., the current unspent outputs) and verifying digital signatures. Hence, the enclave code can maintain the UTXO set and verify the needed signatures for the new blocks. In fact, if the proof-of-stake protocol requires blocks to contain a commitment to the UTXO set, then the complexity of the enclave code will be quite minimal.

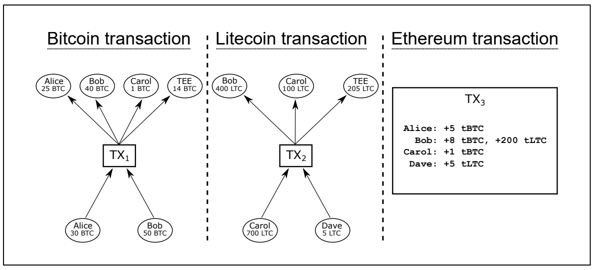


Figure 17: Atomic issuance of tokenized coins.

E FUNGIBLE TOKENIZED COINS

The Tesseract platform also allows its users to withdraw and circulate tokenized coins that are pegged to some specific cryptocurrency, with no need to trust a human element and no exposure to markets fluctuations. Essentially, this is done by maintaining a reserve of the pegged cryptocurrency within the TEE enclave, and employing the all-or-nothing protocol (cf. Section 3) to ensure that the enclave remains solvent.

Thus, for example, Carol can deposit 600 LTC to the Tesseract exchange, trade the 600 LTC for 2 BTC, and withdraw 2 tokenized BTC (tBTC) into the Ethereum blockchain. Then, Carol could deposit her 2 tBTC to any smart contract that recognizes the assets that Tesseract issues. For instance, Carol may wish to play a trust-free poker game in which the pot is denominated in tBTC instead of ETH (it is impractical to play poker directly on the Bitcoin blockchain and instead Ethereum's stateful contracts need to utilized, see [28]). Another example is a crowdfunding contract that raises money denominated in both tBTC and ETH, but returns all the funds to the investors if the target amount was not reached before a deadline.

The issuance of tokenized coins is illustrated in Figure 17. When a user requests to withdraw tokenized coins, the enclave will move the coins to a *reserve* address, and mint the same amount of new tokens (using ERC20 contract, see next). In the illustration:

- Alice withdraws 5 tBTC out of her 30 BTC,
- Bob trades 2 BTC in exchange for Carol's 600 LTC,
- Bob withdraws 8 tBTC and 200 tLTC,
- Carol keeps 1 BTC and withdraws 1 tBTC,
- Dave uses all of his 5 LTC to withdraw 5 tLTC.

The enclave updates its reserve outputs (14 BTC and 205 LTC in the illustration) by adding coin amounts that match the amounts of tokenized coins that the users withdrew.

Unlike the native coin deposits, reserve outputs and the tokenized coins are not constrained by a timeout, and therefore the tokenized coins are fungible. Any holder of tokenized coins (e.g., tBTC) can later deposit her tokens into the enclave (she can create an account on the Tesseract exchange if she does not have one yet), and receive native coins (e.g., BTC) upon doing so. The enclave will simply discard the tokenized coins that were deposited. Hence, the tokenized coins can circulate freely on the blockchain in which they are issued (the Ethereum blockchain in our implementation), without the involvement of the Tesseract exchange.

Given an all-or-nothing settlement (cf. Definition 3.1) for the transaction that moves native coins (from the users to the reserve output) and the transaction that mints tokenized coins, the exchange always remains solvent. In Figure 17 for example, if TX$_1$ is not committed to the Bitcoin blockchain but TX$_3$ is committed to the Ethereum blockchain, then the eventual holders of the 14 tBTC will not be able to deposit their tokens in order to convert them to native BTC, because the reserve output (of 14 BTC) does not exist. Likewise, if TX$_3$ is not committed to Ethereum but TX$_1$ is committed to Bitcoin, then the Bitcoin holders will be damaged (e.g., Alice will lose 5 BTC).

As described in Sections 2 and 3, the all-or-nothing settlement should occur after an interval that is longer than the time that it takes for the all-or-nothing protocol execution to complete (e.g., an interval of 24 hours can be sensible). This means that when a user requests to withdraw tokenized coins, there will be a waiting period (say, somewhere between 1 hour and 25 hours) before she receives the tokens. This also implies good scalability, since all the native coins (that are kept in reserve) are accumulated into a single output that is updated on-chain only after a lengthy time interval.

Since the tokenized coins are issued by the Tesseract exchange and are fungible, the holders of these tokens will be unable to convert them to native coins in the case that the Tesseract platform is destroyed. However, the full protocol (cf. [27]) is distributed and hence less likely to fail. We note that an all-or-nothing settlement is efficient but not essential if the exchange is run in a single enclave without replicas, as it is possible for the enclave to wait for confirmation that the reserve output is increased before creating the signed transaction that issues the new tokenized coins. However, the distributed Π_{RTExch} is useful to retain control over the reserve output even if all but one of the enclaves are destroyed, and the full consensus protocol (which includes the all-or-nothing settlement subroutine) allows us to synchronize the replicas and ensure solvency.

It is also possible to incorporate a timeout to the reserve outputs that specifies that the coins will be controlled by a multi-signature of several reputable parties if Tesseract stops updating the reserve outputs and thus the time expiration is reached. However, this gives an incentive to these parties to destroy the Tesseract platform and collect the reserve coins.

For comparison, cross-chain pegged tokens can also be realized via smart contracts and chain relays, as proposed by the XCLAIM [113] framework. Like Tesseract, this approach does not rely on a privileged group of (supposedly reputable) humans that hold the funds in escrow. However, XCLAIM assumes the existence of an ideal chain relay (e.g., BTC Relay [7]) that verifies and stores all the block headers of one blockchain C_1 on another blockchain C_2. The incentive structure for such a relay service is vague, since all the full nodes of C_2 need to transmit, verify, and store this data (cf. [19]). In Ethereum, a prominent proposal to cope with these costs is to impose a rent surcharge [13], which will in turn increase the cost of BTC Relay. Another difference is that XCLAIM requires one of the two blockchains to have an extensive smart contract support, whereas Tesseract only requires the blockchains to support basic timelocked transactions.