DEXO: A Secure and Fair Exchange Mechanism for Decentralized IoT Data Markets

Yue Li, *Student Member, IEEE*, Ifteher Alom, *Student Member, IEEE*, Wenhai Sun, *Senior Member, IEEE*, Yang Xiao, *Member, IEEE*

***Abstract*—Opening up data produced by the Internet of Things (IoT) and mobile devices for public utilization can maximize their economic value. Challenges remain in the trustworthiness of the data sources and the security of the trading process, particularly when there is no trust between the data providers and consumers. In this paper, we propose DEXO, a decentralized data exchange mechanism that facilitates secure and fair data exchange between data consumers and distributed IoT/mobile data providers at scale, allowing the consumer to verify the data generation process and the providers to be compensated for providing authentic data, with correctness guarantees from the exchange platform. To realize this, DEXO extends the decentralized oracle network model that has been successful in the blockchain applications domain to incorporate novel hardware-cryptographic co-design that harmonizes trusted execution environment, secret sharing, and smart contract-assisted fair exchange. For the first time, DEXO ensures end-to-end data confidentiality, source verifiability, and fairness of the exchange process with strong resilience against participant collusion. We implemented a prototype of the DEXO system to demonstrate feasibility. The evaluation shows a moderate deployment cost and significantly improved blockchain operation efficiency compared to a popular data exchange mechanism.**

***Index Terms*—IoT Data Market, Fair Exchange, Decentralized System, Trusted Hardware.**

I. Introduction

DATA produced by mobile and Internet of Things (IoT) devices are widely seen as valuable assets for the knowledge-based economy, with important applications in mobile network planning, city traffic management, healthcare analytics, and more recently training foundational AI models. Unlike the free data on the Internet, IoT data are usually proprietary and have limited trust boundaries. Sharing these data with consumers from other domains would have profound security and privacy implications. A trusted third party (TTP) such as a data broker is often required to facilitate the data acquisition from distributed providers and the sale of data to interested parties. This model, however, represents a central point of failure and is prone to privacy violations. In an infamous case, T-Mobile was found selling mobile subscribers' location data to third-party data brokers who subsequently sold them to other unauthorized parties, all without subscribers' consent [1]. Similar incidents have occurred in other big telecoms [2], [3] and various data brokers [4].

Ideally, building a data marketplace requires a secure exchange mechanism that facilitates data sales between data providers and data consumers [5]. When it comes to a marketplace for mobile/IoT data, the exchange mechanism faces four specific security challenges. (i) *End-to-end data confidentiality*: the data of interest should only be revealed to the paid consumer in a process transparent to the data provider; it remains confidential to third parties, including the facilitators of the exchange. (ii) *Fair exchange*: the exchange process should ensure that the consumer receives the data only if a payment is made to the provider; the provider receives the payment only if the consumer gets the data. The consumer should also be able to revert/abort the exchange if the data does not meet promised specifications. (iii) *Data source quality and verifiability*: the consumer should receive quality data that conforms to a pre-agreed standard and can be verified for its integrity. (iv) *Resilience*: the exchange process should not suffer from single-point failures; the above goals can be achieved even if a fraction of participants malfunction or collude. Besides security goals, the exchange mechanism should have good *scalability* in data volume, due to the sheer size of data from the distributed mobile/IoT devices.

The recent rise of distributed ledger technology, represented by blockchain, and its native *smart contract* functionality have offered a viable path toward the above vision. Smart contracts allow for the automatic and traceable execution of business logic between untrustful parties with the correctness and liveness enforced by the underlying blockchain consensus. In light of this, there has been extensive research leveraging smart contracts to enable mobile/IoT data marketplaces. One line of research leverages smart contracts as the *on-chain* element of a trusted data broker that facilitates the listing and sale of data [6]–[10]. The data item of interest is usually curated in the broker's *off-chain* server confidentially which can also perform quality control [11], [12]; the contract encodes access control rules that determine the release of the data to the consumer upon receiving a valid bid with payment from the latter. This paradigm bears some similarities to the non-fungible token (NFT) marketplaces [13], [14], where the access control centers around the transfer of data ownership (by utilizing digital signatures) instead of the transfer of data item itself. In both cases, significant trust is placed in the broker's off-chain server for storing and transferring data upon contract rules, posing a risk of single-point failure.

When it comes to data source quality control and verifiabil-

Yue Li, Ifteher Alom, and Yang Xiao are with the Department of Computer Science, University of Kentucky, Lexington, KY, USA (email: {yue.li,ifteheralom,xiaoy}@uky.edu).

Wenhai Sun is with the Department of Computer and Information Technology, Purdue University, West Lafayette, IN, USA (email: whsun@purdue.edu).

ity, a popular data provision paradigm known as *blockchain oracle* provides a potential solution. Blockchain oracles are third-party services designed to help a smart contract procure real-world data critical to its application logic [15]. Compared to data marketplaces, blockchain oracles focus on the data provision process by building a secure channel between the data sources and a consumer contract [16]. The data items are usually collected and curated by a dedicated group of server nodes called the Decentralized Oracle Network (DON) with each node selecting its own data sources [17], [18]. DON nodes aggregate their data on an oracle contract that serves as the query interface to potential consumer contracts. Despite their popularity, existing DONs heavily consolidate the upstream data provision process and can only deliver data on the blockchain. This leads to data diversity and scalability (on-chain cost) challenges that hinder their applicability to data markets [19], [20]. The on-chain data delivery is also restricted to non-confidential data. Nonetheless, DONs provide valuable lessons and established infrastructure for securing the data supply side, which is potentially useful for building a marketplace of verified data.

In this paper, we introduce **DEXO** (**D**ecentralized data **EX**change **O**racle), a new data exchange platform designed to enable a secure and scalable marketplace for mobile/IoT data. DEXO extends the DON model into a decentralized data exchange platform that for the first time accomplishes the security goals of end-to-end data confidentiality, source verifiability, and fair exchange of data with strong resilience to single-point failures. The main infrastructure of the DEXO platform consists of a DON-like node consortium called the **DEXO Network** as the off-chain component and **DEXO Contracts** as the on-chain component. On a high level, the DEXO Network is responsible for the collection and curation of ciphertext data from data providers. A provider-specific DEXO Contract is responsible for data listing and enforcing data access control, fair exchange, and compensation.

On the data supply side, we require owners of IoT/mobile devices that can produce common sensory data to form a data-provider decentralized application (DApp), dubbed P-DApp. The P-DApp represents the data owners in the DEXO data market with a frontend server responsible for collecting and transporting data from each device to the DEXO Network and a dedicated DEXO Contract serving as its on-chain backend, fulfilling data listing and later data owner compensation upon a successful sale. To address the data quality and verifiability challenges, DEXO leverages the emerging availability of trusted execution environments (TEE) in commercial IoT/mobile devices [21], [22] which provides attested execution of sensitive applications. In the data generation stage, DEXO requires each data owner device to instantiate a DEXO-ratified TEE application $\mathcal{F}_{TA}$ that pre-processes raw data and sanitizes them into the required format. $\mathcal{F}_{TA}$ and its output are verifiable for *execution integrity* and *device authenticity* with the help of TEE's attested execution capability [23]. This ensures that the data originates from $\mathcal{F}_{TA}$-equipped devices instead of being mass-generated by unknown sources.

DEXO further achieves end-to-end confidentiality and resilience in the data exchange by integrating secret sharing and a fair exchange mechanism into the data generation and exchange workflow. Besides pre-processing raw data and attaching integrity proofs, $\mathcal{F}_{TA}$ splits the data into N secret shares with each share forwarded through the P-DApp to a specific DEXO node (assuming there are N DEXO nodes). This ensures that each data share, even if being intercepted during transit, remains unintelligible to unauthorized entities and the plaintext data remains confidential to individual DEXO nodes. Only entities possessing a threshold fraction of secret shares (t out of N) can reconstruct the original data.

When a data consumer sends a purchase request to DEXO Contract for a certain data item, each DEXO node will be engaged in a fair exchange process to deliver the corresponding data shares to the consumer. The P-DApp users should receive compensation only if the consumer obtains the correct data at the end of the exchange. This process involves an atomic execution of off-chain delivery of encrypted data shares and on-chain release of the decryption key by utilizing cryptographic commitments [24]. Under the assumption that F out of the N DEXO nodes are compromised with $F < \frac{1}{2}N$ and $F < t \leq N - F$, we prove that the original data remains confidential to individual DEXO nodes at all times and the consumer is guaranteed to reconstruct the data. DEXO also guarantees resilience against potential collision cases in that by colluding with fewer than t DEXO nodes, a consumer cannot scam a P-DApp for any portion of the original data without full payment. Likewise, a P-DApp cannot scam a consumer for payment without providing the full requested data or by colluding with individual nodes who attempt to tamper with the data shares.

To sum up, we make the following contributions:

- We propose DEXO, a new decentralized data exchange mechanism to enable a secure and scalable marketplace for IoT and mobile data. DEXO extends the DON model and for the first time accomplishes end-to-end data confidentiality, source verifiability, fault tolerance, and fair exchange of off-chain data.
- DEXO's data supply side leverages TEE-based data secret sharing to realize confidential and verifiable data procurement from IoT devices. This design is of independent interest to DON services for sourcing sensitive data with origin verifiability.
- DEXO's data exchange side leverages smart contract-based fair exchange for delivering off-chain data from DON nodes to consumers while enforcing payments to original data sources. Our design ensures strong guarantees of fault tolerance and collusion resistance against malicious system participants and also handles disputes between data owners and consumers.
- We provide a proof-of-concept implementation of DEXO, utilizing Ethereum as the smart contract platform and ARM TrustZone as the IoT device TEE platform. The experiment results illustrate that DEXO significantly outperforms existing DON solutions in on-chain gas cost per unit of data consumed, while incurring moderate off-chain execution overhead for individual data providers, demonstrating DEXO efficiency and practicality.

TABLE I: Comparison of Major Data Market Mechanisms and DEXO. Five criteria: (1) Data Source Verifiability—the reliability of a data source can be technically verified; (2) End-to-end Data Confidentiality—data is kept private and not exposed on public platforms like blockchains or non-consumers; (3) Decentralization—preventing system's single-point failure with strong resilience; (4) Fair Exchange of Off-chain Data—allows consumers to obtain refunds if the data does not meet promised specifications; (5) Mobile/IoT Data—support for the exchange of mobile/IoT data.

Scheme	Category	Data Source Verifiability	End-to-end Data Confidentiality	Decentralization (Fault Tolerance)	Fair Exchange of Off-chain Data	Mobile/IoT Data
Ocean Protocol [10]	data exchange	✗	✓	✓	✗	✓
DataBroker DAO [9]	data exchange	✗	✓	✓	✓	✓
OpenSea [14]	NFT market	provenance	✗	✓	ownership only	✗
Rarible [13]	NFT market	provenance	✗	✓	ownership only	✗
FairSwap [25]	data exchange	✗	✓	✗	✓	✓
OptiSwap [24]	data exchange	✗	✓	✗	✓	✓
PrivacyGuard [26]	data exchange	✗	✓	✗	✓	✓
Town Crier [16]	oracle service	✓	✗	✗	✗	✗
DECO [27]	oracle service	✓	✓	✓	✗	✓
Chainlink [17]	oracle service	✓	✗	✓	✗	✗
DEXO	data exchange	✓	✓	✓	✓	✓

The remainder of the paper is organized as follows. Section II discusses the existing work relevant to our scheme. Section III describes the system models and goals. Section IV introduces the building blocks necessary for constructing our scheme. Section V elaborates on the detailed design. Section VI provides security and complexity analyses of our scheme. The implementation and evaluation results are presented in Sections VII-VIII, followed by the conclusion in Section IX.

II. Background and Related Work

A. Smart Contracts and DApps

Smart contracts facilitate the automated and transparent execution of business logic among parties that do not trust each other and are typically instantiated on a distributed ledger or blockchain system [28], [29]. DApps are web applications that leverage blockchain smart contracts for creating decentralized, self-governing, and minimum-trust business logic. A DApp usually comprises three parts: a user interface (such as a browser), a user-facing server as the *frontend*, and a blockchain smart contract as the *backend* [30]. The user interface and frontend server function similarly to traditional web applications, while the smart contract backend is responsible for processing and storing the DApp's main transactional logic whose security is provided by the underlying blockchain consensus. This contrasts with traditional apps whose backend logic is usually handled by a centralized cloud server. DEXO leverages the DApp format to standardize the supply side of the data market by requiring IoT device owners to join data-providing DApps that follow standardized data provision and compensation workflows.

B. Decentralized Data Exchanges

The salient properties of smart contracts also give rise to decentralized data marketplaces that are transparent, auditable, and autonomous. For example, smart contracts can serve as a listing platform for data that are originally stored in an off-chain data broker [6]–[8]. The smart contract can encode certain access and compensation rules that determine the action of the data broker on the release of data to a paid consumer. Ocean Protocol [10] is a decentralized data exchange platform to facilitate data sharing and monetization to unlock data for AI. It relies on its tokenized service layer (with the OCEAN data token) to mediate data exchange. DataBroker DAO [9] targets IoT sensory data, providing a decentralized marketplace for users to buy and sell data generated by IoT devices. It employs smart contracts to facilitate IoT data transactions while keeping records as the contract states. Streamr [31] is a decentralized, peer-to-peer platform for real-time data sharing, concentrating on creating and operating data streams. In comparison to DEXO, these solutions generally do not fulfill data source verification or fair exchange (except DataBroker DAO) as they place trust in the autonomous market participants and data brokers [5].

Another popular type of data exchange is NFT marketplaces. A typical NFT marketplace, such as OpenSea [14] and Rarible [13], takes temporary ownership of the token using an escrow account and the token is transferred to the highest bidder. The exchange process involves the on-chain transfer of ownership where the seller signs the transfer-out invocation with the new owner's account address. It does not involve off-chain fair exchange since the digital object referenced by the NFT is available publicly (at least in partial form). In comparison, DEXO focuses on the value of data itself rather than ownerships. That is, DEXO needs to keep the data confidential from the DEXO marketplace before the buyer provides payment.

C. Smart Contract-based Fair Exchange Protocols

Fair exchange protocols aim to address the lack of trust between the parties of a digital trade. A fair exchange protocol should realize an atomic exchange, meaning the seller is ensured that the buyer can only receive the digital asset when the payment is received, and the buyer is ensured that the seller can only receive the payment when the digital asset is obtained, which together fulfills an atomic exchange. While it is shown that fair exchange is not possible without a TTP

[32], the emergence of blockchain-based smart contracts shows a viable solution by having a smart contract fulfilling the TTP role [24]–[26], [33]–[35].

Two popular smart contract-based fair exchange schemes are FairSwap [25] and OptiSwap [24]. They leverage cryptographic commitments to facilitate the transfer of the buyer's pre-payment to the seller and the release of the decryption key of the data asset to the buyer. In FairSwap [25], the seller initially provides encrypted data along with some auxiliary information to the buyer. The buyer checks the auxiliary information and, if convinced, deposits the money into the smart contract. Once the seller has received an assurance of the payment (locked at the smart contract), the secret key is released to the blockchain, and the buyer is thus able to decipher the witness. FairSwap protocol also employs a Merkle proof-based mechanism called the proof of misbehavior (PoM) concept to deal with invalid witnesses. OptiSwap [24] extends FairSwap by incorporating an interactive dispute resolution protocol executed only in pessimistic cases, thus expediting honestly performed transactions. Specifically, the buyer and seller have a pre-determined predicate function $\phi()$ on the validity of a given data item x ($\phi(x) == 1$ means valid; 0 means invalid). PrivacyGuard [26] achieves a similar goal by using TEE for off-chain data storage and a hash lock mechanism on a smart contract for disclosing the key. While these schemes nicely achieve data confidentiality and fair exchange goals, they do not provide functions for data source quality control or verification and also face risks of single-point failures due to centrally managed data provisioning and storage.

Independent of the above fair exchange schemes, Hash Time-Locked Contract (HTLC) provides another paradigm for exchanging on-chain assets, particularly for enabling cross-chain atomic swaps [36]. HTLC-based atomic swap protocols allow two parties to exchange native cryptocurrencies without relying on trusted third parties and guarantee atomicity—either both receive the cryptocurrencies from each other or neither does. Particularly, due to HTLC's time locking mechanism, if one party does not claim the funds within a specific time, the other party can reclaim the funds and rescind the swap. Recent developments including MAD-HTLC [37] and He-HTLC [38] have addressed HTLC's vulnerability to bribery attacks and other strategic manipulation by cryptocurrency miners, elevating the security of HTLC-based atomic swaps from a game-theoretical perspective. In comparison, fair exchange schemes (including FairSwap [25], OptiSwap [24], and our adaptation) differ from atomic swaps in application scenarios and certain security properties. First, atomic swap is an effective method to exchanging cryptocurrencies (or other on-chain assets) across different blockchains, whereas fair exchange enables the exchange of cryptocurrency for any off-chain digital assets. Second, atomic swap relies on time locks to facilitate graceful exit (required for atomicity), whereas fair exchange relies on cryptographic commitments and custom-defined predicate functions to facilitate a dispute process. Third, fair exchange supports the off-chain delivery of confidential assets, providing a unique advantage for exchanging sensitive data.

D. Blockchain Data Oracles

Blockchain data oracles are third-party services that transport data from external (off-chain) sources into smart contracts. Traditionally, blockchain oracle schemes focus on providing secure channels between smart contracts and external data sources [16], [27], [39], [40]. Town Crier [16] extends Transport Layer Security (TLS) for establishing authenticated communication between HTTPS-enabled websites to client contracts by leveraging TEEs as a trusted intermediary. DECO [27] achieves similar functions through multiparty computation and further provides data confidentiality by using zero-knowledge proofs (ZKP) to validate oracle events without exposing the data in plaintext.

Popular data oracle services such as Chainlink [17] and Band Protocol [18] have offered practical solutions to the data provisioning problem by adopting the decentralized oracle network (DON) model. The DON model stipulates that for every data query (*e.g.,* latest price of a certain asset), a network of independent oracle nodes collect responses from their own selection of sources. To avoid a single point of failure and to reign in the varying quality of data sources, each oracle node aggregates the local responses and submits the result to an on-chain oracle contract. The contract automatically aggregates results from all oracle nodes (*e.g.*, by taking the mean or median) and presents the final result to the consumer. Other DON solutions like WINkLink [41] also utilize reputation mechanisms to promote the honest participation of oracle nodes. Despite their popularity, existing DON solutions face data confidentiality and scalability challenges. The oracle service often requires the plaintext data be publicized and consumed on a smart contract in an open manner, which is not ideal for sensitive data items. At the same time, the data that a DON can feed to the blockchain are often limited in size, mainly due to the blockchain's intrinsic scalability limitation attributed by high on-chain costs for computation and storage [20]. Each oracle node in a DON also needs to keep a list of "premium sources" and only collects data from them to maintain the quality of its data offering (*i.e.*, the consumers) [42]. Consequently, nodes in the DON tend to gradually converge on selecting from a limited list of well-known data sources. This tendency has posed another significant constraint on DONs' applicability to enabling data marketplaces.

DEXO shares similarities with DON solutions by relying on decentralized oracle nodes to mitigate single-point failures. However, traditional oracle services including DONs transmit data to smart contracts on the blockchain, which provides no data confidentiality and limits data volume and efficiency. In contrast, DEXO transmits data off-chain with on-chain settlement for exchange, achieving end-to-end confidentiality and greater scalability in data sizes. Additionally, while DONs rely on node operators to determine their own data sources, DEXO allows distributed data sources to participate in the data market directly by joining P-DApps. This shift reduces reliance on monopolistic data providers and fosters a more diverse data marketplace.

TABLE II: List of Notations

Notation	Description
N	Number of DEXO nodes
$\mathcal{N}_i$	DEXO node i ($i \in [N]$)
$\mathcal{C}_{DEXO}$	DEXO contract
F	Maximum compromised DEXO nodes ($F < \frac{1}{2}N$)
t	Secret sharing threshold ($F < t \leq N - F$)
$\mathcal{F}_{TA}$	Trusted application within a TEE
$\mathcal{F}_{sc}$	Smart contract ideal functionality
$\mathcal{G}_{att}$	TEE attested execution functionality
$\mathcal{F}_{ss}$	Secrete sharing functionality
$\mathcal{F}_{com}$	Commitment ideal functionality
D_i	Raw data gathered by P-DApp user i
d_i	Pre-preprocessed and formatted data for trading
$ds_{i,j}$	Secret data share of user i for DEXO node j
$\sigma_{i,j}$	$\mathcal{F}_{TA}$'s signature over and $ds_{i,j}$ and TEE runtime
cid	Contract identifier for data exchange
eid	TEE instance identifier
mpk	Master public key for TEE attestation
msk	Master secret key for TEE attestation
Δ_j	Merkle tree root hash of encrypted shares submitted by node j
z	Encrypted data shares sent to consumers
k_j	Secret key used by node j to encrypt data shares

III. System Model

A. Participation Model

We define four participant types in the DEXO data market:

- **Data-providing DApp** (**P-DApp**, or **provider**) is a seller in the data market. It advertises the availability of certain *off-chain* user-end data through DEXO and expects to receive compensation once DEXO facilitates a sale of the advertised data. A P-DApp is a DApp, comprising of a backend smart contract (called a DEXO contract) and a user-facing frontend server.
- **P-DApp users** are P-DApp's end users, typically mobile or IoT device owners, who agree to participate in the P-DApp's data sales through DEXO. They are data sources of the P-DApp and will be compensated for data sales. P-DApp users can interact with P-DApp's frontend server through standard secure communication protocols such as TLS.
- **Consumer** is a buyer in the data market. A consumer can browse a P-DApp's data advertisements on DEXO and is willing to pay for an interested data item.
- **DEXO node** is reminiscent of an oracle node in existing DON schemes (*e.g.*, Chainlink). A fixed number of DEXO nodes constitute the **DEXO network** that jointly fulfills the data exchange mission between P-DApps and consumers. We assume there are fixed N DEXO nodes in our system.

Besides the above roles, we assume that a **smart contract platform**, such as Ethereum, is in place to serve as the backend environment of P-DApps and the DEXO network. We further assume a P-DApp prices its data items. It creates a smart contract that specifies the data description and predefined price prior to any exchanges with a consumer through DEXO. DEXO acts as a neutral platform that facilitates the exchanges and does not intervene in the pricing process. We leave more complex pricing schemes, such as auctioning, to future work.

B. Design Goals

DEXO aims to enable decentralized data exchange with the following objectives:

O1: Data source verifiability. To ensure the quality of the data collected from sources, DEXO requires each P-DApp user to pre-process locally gathered data using a provided function. This pre-processing function ensures the data for sale conforms to a certain format and normality as advertised by the P-DApp and should be verifiable for integrity by the DEXO network. In this work, we do not require DEXO to provide broader data-derived quality control, such as assessing the contextual utility of the data. We leave such considerations to data consumers who can decide to purchase future data from the P-DApp.

O2: End-to-end data confidentiality. The data requested by a consumer should only be revealed to the consumer; it remains confidential to the DEXO nodes and the public.

O3: Fault tolerance and no single-point failure. The DEXO network is decentralized, with multiple nodes working to facilitate the data verification and exchange process. The above objectives should still be accomplished when a minority of nodes are compromised and do not follow the correct protocol.

O4: Fair exchange. DEXO should facilitate a fair exchange between a P-DApp and a consumer when the latter requests the former's data. The P-DApp should receive compensation only if the consumer obtains the correct data, and vice versa.

O5: Off-chain data delivery. DEXO should facilitate a fast exchange process for off-chain data delivery and on-chain verifiability. The blockchain costs should be minimized. This is a key difference from the existing DON schemes where the entirety of requested data has to be delivered on-chain, constraining the data size due to the on-chain cost.

In particular, realizing the objectives requires handling faulty behaviors and potential collusion among the participants. The detailed threat model is provided in Section III-C.

C. Threat Model

Out of the N nodes in the DEXO network, we assume at most $F < \frac{1}{2}N$ of them are compromised at any point and the rest will operate correctly. This threshold assumption is sufficient to encompass that of existing oracle networks such as Chainlink [42] where at most 6 out of the 21 oracle nodes could be compromised. Specifically, the malicious activities of a compromised node relevant to DEXO's operation include:

1) not following the designated protocol and sending arbitrary information to other participants in the system;
2) extracting a P-DApp user's data shares and exposing them in untrusted domains;
3) colluding with a P-DApp to scam a consumer for payment without providing the requested data in full;
4) colluding with a consumer to scam a P-DApp for any portion of useful data without a full payment.

The last two colluding situations imply that the P-DApp or consumer may not execute a given data exchange protocol faithfully. In light of this, DEXO should be able to allow either party to abort the exchange safely.

For the data source aspect, we assume the P-DApp users are responsible for managing their own raw data source. They will be ultimately compensated for providing high-quality data. We require that a user's locally established TEE is tamper-proof and performs attested execution of certain data pre-processing rules required by the DEXO system. Specifically, The TEE safeguards the pre-processed data by isolating it within a secure enclave, preventing unauthorized access or tampering even from the host operating system. The TEE attestation provides proof of TEE program's integrity and TEE hardware's authenticity. Furthermore, the trusted application $\mathcal{F}_{TA}$ in TEE is ratified by the DEXO community and available in the public domain. It should always correctly execute the data pre-processing and security functions (*e.g.*, secret sharing and generating attestation reports) following the standard TEE security properties. We also assume the P-DApp server is trusted by its users for not leaking their data before the sale. Lastly, the TEE attestation service is trusted for verifying the TEE platform's authenticity when it receives an attestation report forwarded by a DEXO node.

IV. Building Blocks

In this section, we describe the building blocks of DEXO, including their key properties and ideal functionalities. Using the ideal functionalities allows us to abstract away their implementation and focus on composing the DEXO system.

A. Smart Contract

Smart contracts enable the automatic and traceable execution of multiparty business logic and typically live in an append-only blockchain ledger. We adopt the ideal blockchain functionality $\mathcal{F}_{blockchain}$ proposed in [43] as the baseline. It allows participants to read on-chain information through the $\mathbf{read}()$ interface and append new information through the $\mathbf{write}()$ interface. Here we describe an ideal smart contract functionality $\mathcal{F}_{sc}$ by extending $\mathcal{F}_{blockchain}$ to incorporate more expressive contract operations as follows:

Definition 1 (Ideal Functionality $\mathcal{F}_{sc}$): The ideal smart contract functionality $\mathcal{F}_{sc}$ inherits $\mathcal{F}_{blockchain}$'s persistent storage $\mathrm{LStorage}$ and supports the following interfaces:

- $\mathcal{F}_{sc}.\mathbf{create}(\mathcal{C}, params[])$ creates a smart contract with a given contract encoding $\mathcal{C}$ and initializing parameters $params[]$. If successful, it generates a contract identifier cid, writes the contract object into $\mathrm{LStorage}$, and returns SUCCESS and cid to the sender.
- $\mathcal{F}_{sc}.\mathbf{read}(cid, \text{“}var\text{”})$ looks up the smart contract identified by cid and the state variable identified by *var* within $\mathrm{LStorage}$. If *var* exists, it returns the variable value to the sender.
- $\mathcal{F}_{sc}.\mathbf{write}(cid, \text{“}func\text{”}, args[])$ writes to the smart contract identified by cid invoking the specified function identified by *func* with arguments $args[]$. If successful, it updates the modified state variables in $\mathrm{LStorage}$ and returns SUCCESS to the sender.

B. TEE-based Attested Execution

TEE is a secure area within a processor that provides an isolated and protected environment for executing sensitive code and handling confidential data. TEEs are designed to ensure the confidentiality, integrity, and authenticity of the data and code running within them even amid malicious software or hardware attacks on the hosting system [44]. Popular TEE platforms include Intel Software Guard eXtensions (SGX) [45], AMD Secure Encrypted Virtualization (SEV) [46], ARM TrustZone [21], and Apple Secure Enclave in T2 chip [22], showing a diverse ecosystem of TEE implementations on various CPU architectures. A key functionality of TEE is attested execution that safely executes the TEE program while proving the program's authenticity and integrity. A signature for the enclave is created by using a hard-coded key based on the TEE initial state, code, and data and then verified with the help of chip vendors [47], [48]. In this paper, we assume that TEE capability is available for mobile and IoT devices and adopt the generalized attested execution functionality $\mathcal{G}_{att}$ defined in [49] (while ARM TrustZone is used for experiments). Here we provide a simplified description of $\mathcal{G}_{att}$:

Definition 2 (Ideal Functionality $\mathcal{G}_{att}$): $\mathcal{G}_{att}$ is the ideal functionality of general TEE-based attested execution. It is hard-coded with a public-private key pair (mpk, msk), keeps a persistent TEE memory TMem, and provides the following interfaces:

- $\mathcal{G}_{att}.\mathbf{install}(\mathcal{F}_{TA})$ establishes a new TEE instance inside TMem from the the caller-provided trusted application $\mathcal{F}_{TA}$. $\mathcal{F}_{TA}$'s state variables are also stored in TMem. If successful, it generates an identifier eid for the new TEE instance and returns eid to the caller.
- $\mathcal{G}_{att}.\mathbf{resume}(eid, args[])$ executes the program (*i.e.* $\mathcal{F}_{TA}$) inside the eid-identified TEE instance with the given arguments $args[]$. If successful, it returns the execution result res and a signature over the TEE runtime σ^{rt} signed by msk. Any modified state variables are updated in TMem.

The $\mathcal{G}_{att}.\mathbf{resume}()$ essentially fulfills the attested execution functionality with σ^{rt} attesting to the authenticity and integrity of the TEE program $\mathcal{F}_{TA}$. TEE attestation ensures that all data processed by the TA remains confidential and tamper-proof. The TEE safeguards the data by isolating it within a secure enclave, preventing unauthorized access even from the host operating system or hardware. The attestation mechanism generates a signed report to verify that the data was handled securely by the TA, ensuring that no unauthorized modifications occurred during processing. We will describe a detailed $\mathcal{F}_{TA}$ and the $\mathbf{resume}$ procedure in Section V-B.

C. Shamir's Secret Sharing

Secret sharing is a cryptographic technique to distribute a confidential datum d into multiple (n) fragments, known as shares. A (t, n)-secret sharing scheme ensures that anyone with no fewer than t of the shares can reconstruct d. for which we describe the ideal functionality as follows:

Definition 3 (Ideal Functionality $\mathcal{F}_{ss}$): The ideal secret sharing functionality $\mathcal{F}_{ss}$ provides two interfaces:

- $\mathcal{F}_{ss}.\mathbf{createshares}(t, n, d)$ generates n shares from the provided secret d so that any t out of the shares can be used to reconstruct d.
- $\mathcal{F}_{ss}.\mathbf{reconstruct}(t, n, ss[])$ either returns the secret d successfully reconstructed from the set of shares $ss[]$ or an error indicating there are inconsistencies in $ss[]$ that preventing the reconstruction of a single secret.

In DEXO, Shamir's Secret Sharing [50] is used to achieve end-to-end data confidentiality and resist single-point failures. We disperse the data into multiple shares which are provided to different DEXO nodes. It ensures any smaller subset of shares below this threshold remains oblivious to any useful knowledge of the original data so that even if a subset of nodes falls prey to security breaches, the integrity and confidentiality of the data remain untarnished.

D. Cryptographic Commitment

A cryptographic commitment allows one to commit to a message while keeping it hidden from other parties, with the ability to open the committed message later [51]. It crucially achieves the binding property—the committing party cannot claim a different message was committed. We adopt the general single-message commitment functionality $\mathcal{F}_{com}$ defined in [52] and provide a concise description as follows:

Definition 4 (Ideal Functionality $\mathcal{F}_{com}$): The ideal single-message commitment functionality $\mathcal{F}_{com}$ keeps a persistent commitment storage $CStorage$ provides the following interfaces:

- $\mathcal{F}_{com}.\mathbf{commit}(sid, P_i, P_j, msg)$ allows the caller P_i to generate a commitment generates a commitment com to msg of sequence number sid. If successful, it stores (sid, com) in $CStorage$, forwards msg to P_j (if P_j is specified), and return SUCCESS to P_i.
- $\mathcal{F}_{com}.\mathbf{open}(sid, P_i, P_j)$ allows the caller P_i to open its previous commitment identified by sid from $CStorage$ and discloses the original msg to P_j (if P_j is specified).

The commitment scheme has been used for constructing fair exchange protocols [24], [25]. In DEXO, we employ a similar construction of fair exchange protocol as in OptiSwap [24] with a modification that a DEXO node commits to each data secret share rather than the original data.

V. DEXO Design

A. System Overview

Achieving the objectives outlined for DEXO requires addressing several unique design challenges that existing DONs or decentralized data exchange solutions fail to tackle. We integrate TEE-based secret sharing with smart contract mechanisms to enable end-to-end data confidentiality while maintaining data verifiability. This integration ensures secure handling of sensitive data without exposing plaintext to any intermediary, which commercial DONs do not achieve. We also design protocols to achieve fault tolerance in the exchange process, allowing the system to handle misbehavior by up to F compromised DEXO nodes while still delivering data successfully to consumers. In contrast, existing decentralized data exchange solutions rely on trusted intermediaries or single-point designs that remain vulnerable to such faults.

Now we describe DEXO's architectural design and high-level workflow, as shown in Fig. 1. To simplify the description, we consider the case of one P-DApp and one consumer. We assume the P-DApp has M end users who can contribute local data to the P-DApp's data sale. We assume there are N DEXO nodes that are pre-determined, denoted $\mathcal{N}_1, ..., \mathcal{N}_N$.

Stage 0: Initialization—This stage aims to set up the foundational components for secure data exchange, including configuring the P-DApp server, enabling TEE functionality on user devices, and deploying the DEXO Contract for fair exchanges.

To become a seller (*i.e.*, P-DApp) in DEXO's data exchange market, a DApp needs to make certain architectural modifications on both its frontend server and user ends. The P-DApp server can establish TLS connections with all DEXO nodes to forward user data to the DEXO network. We assume each user can establish a secure TLS connection with the P-DApp server as in most web applications.

Each user who is willing to participate in the data offering needs to enable the TEE functionality on their device. They establish a new TEE container to instantiate a **trusted application** $\mathcal{F}_{TA}$ which is ratified by DEXO and available in the public domain. $\mathcal{F}_{TA}$ performs data processing and generation of data shares in Stage 1 as we will describe shortly. Once $\mathcal{F}_{TA}$ is instantiated in a TEE container, its program integrity can be verified by the server and any DEXO node through remote attestation.

The P-DApp server creates a dedicated **DEXO Contract** $\mathcal{C}_{DEXO}$ that follows a pre-defined format (see §V-E) and designates the DEXO notes as curators. $\mathcal{C}_{DEXO}$ acts as an adjudicator of fair exchange, holding the payment and ensuring its transfer to the P-DApp users if the exchange is completed or returning it to the consumer if the correct data is not received. We assume the DEXO nodes have access to the attestation service for each type of TEE platform. Attestation services are normally provided by the TEE vendors and are generally assumed to behave honestly.

Stage 1: Data Production—This stage aims to process raw data from P-DApp users, generate secure and verifiable data shares using TEE, and initialize a smart contract with the data description and pricing information for exchange. This stage involves the server and users of a P-DApp. A P-DApp user i gathers raw data D_i and feeds them into the TEE program $\mathcal{F}_{TA}$. $\mathcal{F}_{TA}$ performs the following tasks:

- **Pre-processing:** D_i is processed per a given rule (e.g., maximum value range, moving average) and converted into a certain format. The formatted result is denoted d_i.
- **Secret Sharing:** d_i is split into N shares via a (t, N)-secret sharing algorithm, with each share denoted $ds_{i,j}$ for $j \in [N]$. One can reconstruct d_i with at least t shares. We require $F < t \leq N - F$ to ensure a safe reconstruction of d_i by the consumer (to prove in Section VI).
- **Signature Generation:** Following the data production, a

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2025.3535671

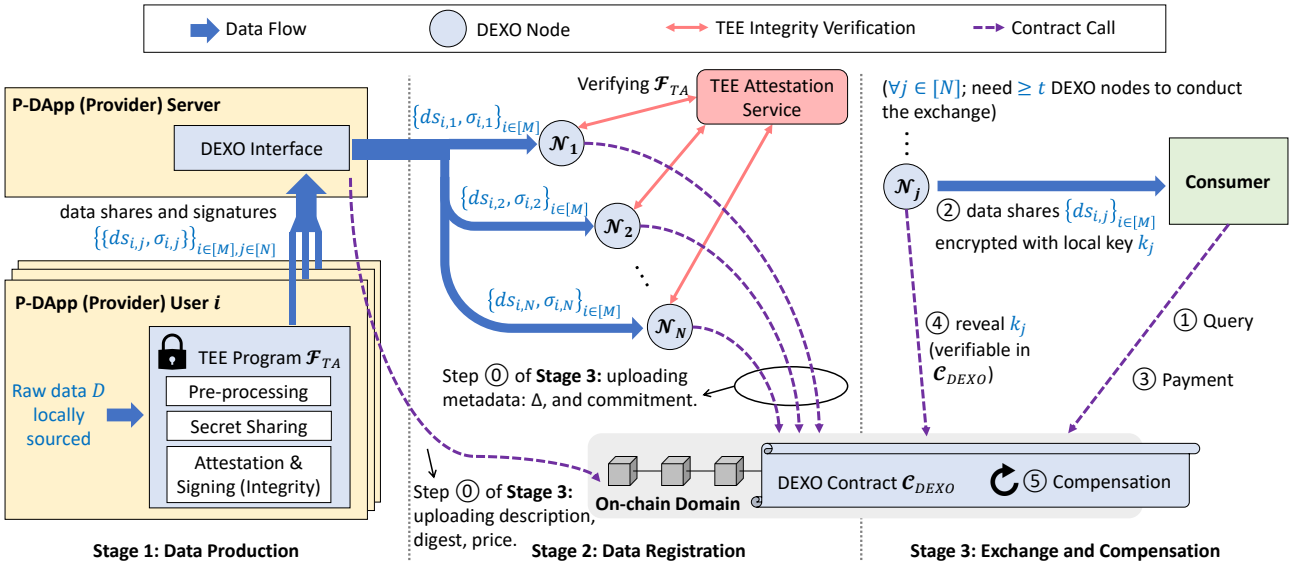


Fig. 1: DEXO System Architecture and Workflow

signature $\sigma_{i,j}$ is created based on each share $ds_{i,j}$ and the TEE runtime measurement with the TEE platform private key. This signature proves the integrity of the data share generation and the authenticity of the TEE platform.

When $\mathcal{F}_{TA}$ finishes, the P-DApp Server creates a data item on the smart contract, initialized with a description $\{desc\}$, along with the price for that data. The P-DApp server also sends addresses of data sources and DEXO nodes to the contract. This process also serves as Step ⓪ for the exchange protocol in Stage 3 together with the "Ready to Exchange" step in Stage 2.

Stage 2: Data Registration—This stage aims to verify the authenticity of data shares through attestation, prepare encrypted shares for exchange, and register the metadata and cryptographic commitments with the DEXO Contract.

Upon receiving the j^{th} shares and signatures from all M users of the P-DApp, *i.e.*, $\{ds_{i,j}, \sigma_{i,j}\}$ for $i \in [M]$, DEXO node $\mathcal{N}_j$ performs the following steps:

- **Source Attestation:** $\mathcal{N}_j$ verifies the authenticity and integrity of data share $ds_{i,j}$ as well as user i's TEE platform with the help of an attestation server. This step ensures $ds_{i,j}$ can be only accepted if it is produced by the required $\mathcal{F}_{TA}$ on an unaltered TEE platform.
- **Ready to Exchange:** After source attestation, $\mathcal{N}_j$ needs to use data shares to prepare for the exchange transaction. $\mathcal{N}_j$ first generates a local secret key k_j and encrypts the local data shares $\{ds_{i,j}\}_{i\in[M]}$ to get the ciphertext z_j. A data digest Δ_j is generated from z_j and so is a cryptographic commitment com_j on k_j. $\mathcal{N}_j$ then submits Δ_j, com_j to $\mathcal{C}_{DEXO}$ in one contract call. The $\{\Delta_j, com_j\}$ submitted by all $j \in [N]$ constitute the metadata for P-DApp. This process also serves as the Step ⓪ for the exchange protocol in Stage 3.

Stage 3: Exchange and Compensation—This stage aims to facilitate the fair exchange of data for payment, ensure data reconstruction integrity, handle potential disputes, and distribute compensation to P-DApp users.

- **Fair Exchange:** It is started by the consumer with a query to the $\mathcal{C}_{DEXO}$ based on the data description $desc$ (Step ①) and establish a TLS connection with each DEXO node. Upon observing the query on-chain, node $\mathcal{N}_j$ ($\forall j \in [N]$) sends the ciphertext z_j to the consumer (Step ②). Then the consumer verifies the integrity of z_j with the corresponding on-chain metadata (data shares digest) before calling to $\mathcal{C}_{DEXO}$ to indicate acceptance with payment (Step ③). Then $\mathcal{N}_j$ calls $\mathcal{C}_{DEXO}$ to reveal its local key k_j, whose integrity can be automatically verified within $\mathcal{C}_{DEXO}$ (Step ④). Once verified, consumer can retrieve k_j from $\mathcal{C}_{DEXO}$ and use it to decrypt z_j to get $\{ds_{i,j}\}_{i\in[M]}$.
- **Data Reconstruction and (Optional) Dispute Handling.** Once decrypting the shares, the consumer can reconstruct the original data. If both parties conduct their operation with integrity, this stage is unnecessary for the buyer or seller. However, when the consumer finds that the shares cannot be used for reconstruction or the reconstructed data item does not conform to the promised attributes specified in the contract, the buyer can initiate a dispute. We provide more details of dispute handling in Section V-E.
- **Provider Compensation:** If all goes successfully (no dispute), $\mathcal{C}_{DEXO}$ distributes the consumer's payment to the P-DApp's data-providing users, fulfilling the final compensation (Step ⑤).

In what remains of this section, we describe each participant routine in the $(\mathcal{G}_{att}, \mathcal{F}_{sc}, \mathcal{F}_{ss}, \mathcal{F}_{com})$-hybrid model based on the Universal Composability (UC) framework [53], utilizing the ideal functionalities described in Section IV.

B. P-DApp User Routine

As the distributed data sources for DEXO, P-DApp users need to execute the DEXO-ratified $\mathcal{F}_{TA}$ for pre-processing raw data and generating data shares. The P-DApp user routine is shown in Algorithm 1. Once $\mathcal{F}_{TA}$ is installed on the user device $\mathcal{G}_{att}.\mathbf{install}(\mathcal{F}_{TA})$, the server can attest to $\mathcal{F}_{TA}$'s integrity by sending an ATTEST command and then collect data from the user by sending a SOLICIT command. Once solicited, user i resumes to the $\mathcal{F}_{TA}$ at the GENDATA command, which produces the secret shares $\{ds_{i,j}\}_{j\in[N]}$ and corresponding signatures $\{\sigma_{i,j}\}_{j\in[N]}$. It is worth noting that $\sigma_{i,j}$ results from signing $ds_{i,j}$ and the runtime measurement with the TEE platform private key. The secret shares and signatures are sent back to the P-DApp server, which disseminates them to the different DEXO nodes accordingly. Each share's signature accompanies it as it moves through the system. To ensure the appropriate distribution of secret shares, the TEE program $\mathcal{F}_{TA}$ also specifies the destination DEXO node for each share during its generation, for instance, share $ds_{i,j}$ is destined to node j. The P-DApp server then relays the shares to the specified nodes. For secure data transmission, the P-DApp server can leverage standard secure communication mechanisms to deliver the data shares to DEXO nodes, such as establishing a TLS connection with each DEXO node which acts as a public web server.

When the P-DApp server receives data from various data sources, it needs to prepare certain information to construct the smart contract. This information includes $desc$, price, and addresses of data sources and DEXO nodes.

Algorithm 1 P-DApp User i Routine

Parameters: t, N

/* Normal Routine (insecure world) */

On init:
- $eid \leftarrow \mathcal{G}_{att}.\mathbf{install}(\mathcal{F}_{TA})$;

On receive ("ATTEST") from $Server$**:**
- $(rt\_msmt, \sigma^{rt}) \leftarrow \mathcal{G}_{att}.\mathbf{resume}(eid, \text{"ATTEST"})$;
- Send ("ATTREPORT", rt_msmt, σ^{rt}) to $Server$;

On receive ("SOLICIT") from $Server$**:**
- Gather rawdata D;
- $(\{ds_i\}, \{\sigma_i\}, mpk_i) \leftarrow \mathcal{G}_{att}.\mathbf{resume}(eid, \text{"GENDATA"}, N, D)$;
- $\forall j \in [N]$: send ("DATASHARE", j, ds_j, σ_j, mpk_i) to $Server$;

/* Trusted Application $\mathcal{F}_{TA}$ (the TEE program, executed upon $\mathcal{G}_{att}.\mathbf{resume}()$) */

On resume $(eid, args[])$**:**
- Generate TEE runtime measurement rt_msmt;
- Retrieve TEE platform public-private key pair (mpk, msk);
- $\sigma^{rt} \leftarrow \mathbf{Sign}_{msk}(rt\_msmt)$; // signature of runtime env.
- **if** ("ATTEST" $\in args[]$) **then**
  - **return** $(rt\_msmt, \sigma^{rt}, mpk)$;
- **end**
- **if** ("GENDATA" $\in args[]$) **then**
  - Read N, D from $args[]$;
  - Pre-process D and convert it to the required format, get d;
  - $(ds_1, ds_2, ..., ds_N) \leftarrow \mathcal{F}_{ss}.\mathbf{createshares}(t, N, d)$;
  - $\sigma_j \leftarrow \mathbf{Sign}_{msk}(ds_j, rt\_msmt)$;
  - **return** $(\{ds_j\}_{j\in[N]}, \{\sigma_j\}_{j\in[N]}, mpk)$;
- **end**

C. DEXO Node Routine

DEXO incorporates attestation and verification mechanisms to establish trust in user-provided data. As is shown in Algorithm 2, DEXO nodes bear the responsibility of managing the participation of DApps in the system and verifying the integrity of data shares provided by each P-DApp user.

Algorithm 2 DEXO Node Routine

/* Data source verification */

On receive ("DATASHARES", $\{ds\}, Signatures, pubkeys, desc, userIDs$) from a P-DApp $Server$**:**
- Send ("VERIFYTEE", $tid, pubkeys$) to $Attestation\ Server$;
- Store $DS[tid] \leftarrow (\{ds\}, desc, userIDs)$;

/* Data Publication */

On receive ("TEEVERIFIED", tid) from $Attestation\ Server$**:**
- Generate secret key k;
- $x[] \leftarrow DS[tid]$;
- $z[] \leftarrow \mathbf{Encrypt}_k(x[])$;
- $\Delta \leftarrow \mathbf{MTHash}(z[])$; //Merkle tree root hash
- $com \leftarrow \mathcal{F}_{com}.\mathbf{commit}(tid, self, \_, k)$;
- $\mathcal{F}_{sc}.\mathbf{write}(cid, \text{"}initialize\text{"}, tid, seller, price, desc, \Delta, com)$;

/* If a buyer has queried the data */

On receive ("NOTICEBUY", $cid, BuyerID$)) from $Buyer$**:**
- $status \leftarrow \mathcal{F}_{sc}.\mathbf{read}(cid, \text{"}buyerStatus.BuyerID\text{"})$;
- If $status =$ QUERIED: Send $z[]$ to $Buyer$;

/* If $z[]$ is accepted */

On receive ("NOTICEACCEPT", $cid, BuyerID$)) from $Buyer$**:**
- $status \leftarrow \mathcal{F}_{sc}.\mathbf{read}(cid, \text{"}buyerStatus.BuyerID\text{"})$;
- If $status =$ ACCEPTED: $\mathcal{F}_{sc}.\mathbf{write}(cid, \text{"}revealKey\text{"}, k)$;

Specifically, node $\mathcal{N}_j$ shall receive data shares $\{ds_{i,j}\}_{i\in[M]}$ and signatures $\{\sigma_{i,j}\}_{i\in[M]}$ if the P-DApp has M users to provide data. For user i's data, $\mathcal{N}_j$ needs to verify the data generation with the help of an attestation server. Upon receiving $\sigma_{i,j}$ and mpk_i of user i, the attestation server can verify the authenticity of user i's TEE platform (*i.e.*, whether it is a genuine TEE hardware) by checking against the public key certificate for mpk_i. Once validated, the attestation server notifies $\mathcal{N}_j$ of the success and the latter can then use mpk_i to validate the signature $\sigma_{i,j}$. If validated, $\mathcal{N}_j$ knows that the data share $ds_{i,j}$ has not been tampered with and can safely proceed to publicize $\{ds_{i,j}\}_{i\in[M]}$ onto $\mathcal{C}_{DEXO}$. $\mathcal{N}_j$ then encrypts $\{ds_{i,j}\}_{i\in[M]}$ with a newly generated secret key k_j, resulting in ciphertext z_j. It also computes a cryptographic digest of z_j, denoted Δ_j. It can be conveniently fulfilled by the Merkle tree root ($\mathbf{MTHash}$) of all fixed-size fragments of z_j. The key k_j is fed to the commitment scheme $\mathcal{F}_{com}.\mathbf{commit}$ so that the result com_j can later be opened to verify the integrity of k_j. It performs step ⓪ of the data exchange, by uploading com_j, Δ_j to the $\mathcal{C}_{DEXO}$. At this point, node $\mathcal{N}_j$ is ready to process data queries from consumers.

Once a consumer indicates an interest in the P-DApp's data (step ①), $\mathcal{N}_j$ will need to provide its encrypted data shares z_j to the consumer (step ②). Upon receiving an initial payment from the consumer to $\mathcal{C}_{DEXO}$ (③), $\mathcal{N}_j$ then discloses the secret key k_j in plaintext to $\mathcal{C}_{DEXO}$, where k_j will be validated by opening the previous commitment com on the (④). A success will start a count-down from

a $timeout$ value (included in $desc$) which provides a buffer for the consumer to submit any dispute. When the timeout passes, the payment will be automatically redistributed to the P-DApp users as compensation. Interactive dispute handling is described in Section V-E.

D. Consumer Routine

The consumer routine is shown in Algorithm 3. It requires the consumer to actively observe existing DEXO contracts for any data of interest. Once determined the target contract $\mathcal{C}_{DEXO}$ and data item indexed by $dataID$, the consumer retrieves the corresponding $desc_j$ and the data shares digest Δ_j for all possible $j \in [N]$. To declare their intent, the consumer submits a query to $\mathcal{C}_{DEXO}$ using their cryptocurrency account address (step ①). This wallet address is recorded in the contract for later payment processing and ensures secure identification throughout the exchange. The query can also provide optional user identifiers, such as IP address or email, for future auditing purposes. Then the consumer connects to at least t DEXO nodes (we use $\mathcal{N}_{[t]}$ to denote the set of their indices) and requests for their encrypted data. This requires the consumer to establish a Web connection (usually over TLS) as a client with each DEXO node and pass authentication for the ownership of its cryptocurrency account address. This process is similar to the interaction between existing Web3 frontend servers and Web3 users, where third-party account management software (such as MetaMask and Alchemy) can be integrated into the Web interface to facilitate account authentication.

Once receiving z_j from $\mathcal{N}_j$, the consumer verifies its integrity against Δ_j, by recomputing the Merkle hash root. If it is verified, the consumer calls $\mathcal{C}_{DEXO}$'s $accept$ function with the required payment (step ③). The payment is now temporarily held in $\mathcal{C}_{DEXO}$'s balance. Once $\mathcal{N}_j$ releases k_j to $\mathcal{C}_{DEXO}$ and the verification with commitment passes (step ④), the consumer can safely obtain k_j to decrypt the corresponding z_j to get $\{ds_{i,j}\}_{i\in[M]}$). When t keys are obtained, the consumer can finally reconstruct the original data d_i from $\{ds_{i,j}\}_{j\in\mathcal{N}_{[t]}}$) by using $\mathcal{F}_{ss}.\mathbf{reconstruct}(t, n, \{ds_{i,j}\}_{j\in\mathcal{N}_{[t]}})$.

E. Fair Exchange with DEXO Contract

The previous sections have described the basic process of a fair exchange protocol session between a DEXO node and a consumer. In DEXO, we employ a parallel composition of OptiSwap [24] for a fair data exchange process between a consumer and the DEXO oracle nodes (each of who has a secret share of the consumer's requested data) with the help of the DEXO contract $\mathcal{C}_{DEXO}$ (see Algorithm 4). All parties are aware of a predicate function $\phi()$ that involves secret reconstruction to determine the misbehavior of individual DEXO nodes. First, $\mathcal{C}_{DEXO}$ serves as the public storage of a P-DApp's metadata, *i.e.*, $\{desc_j, \Delta_j, com_j, \}_{j\in[N]}$. This allows the consumer to verify the ciphertext data from the DEXO nodes (Step ②) and also serves as the temporary custodian of the payment (Step ③). It allows the opening of the commitment com_j in one contract call (Step ④). If the opening succeeds, the j^{th} portion of consumer payment is considered finalized.

Algorithm 3 Data Consumer Routine

```
/* If a new transaction is found                                   */
On receive ("NoticeNewData", cid, dataID) from Self:
    nodeID, price, auxInfo, Δ, dataSources);
    (nodeID, price, auxInfo, Δ) ←
    F_sc.read(cid, "dataSources.dataID");
    If satisfied: F_sc.write(cid, "query'', dataID, buyerID);
/* Accept encrypted shares from selling node                       */
On receive ("EncryptedShares", z[], cid, dataID) from N_j:
    Recompute Δ' ← MTHash(z[]);
    (nodeID, price, auxInfo, Δ) ←
    F_sc.read(cid, "dataSources.dataID");
    If Δ = Δ': F_sc.write(cid, "accept'', price, buyerID);
/* Accept key                                                      */
On receive ("NoticeKey", cid, dataID) from N_j:
    k ← F_sc.read(cid, "keyRevealed.dataID");
    Decryption: x[] ← Decrypt_k(z[]);
    {ds_{i,j}}_{i∈[M]} ← x[];
/* Once more than t keys are obtained from DEXO
   nodes (node set: N_[t])                                         */
On receive ("Reconstruct") from Self:
    Get t, N and timeout from the auxInfo obtained from the
    previous dataSources reading;
    For i ∈ [M]: d_i ← F_ss.reconstruct(t, n, {ds_{i,j}}_{j∈N_[t]});
/* Optional dispute procedure                                      */
On receive ("Dispute") from Self:
    /* Case 1: Reconstruct Original Data and
       Validate Against Description                                */
    Initialize valid = False;
    Select initial sets S_1, S_2 with t + 1 shares each;
    While valid == False:
        d_1 ← F_ss.reconstruct(t, n, S_1);
        d_2 ← F_ss.reconstruct(t, n, S_2);
        If d_1 == d_2:
            valid = True; // Data consistency confirmed
            Set d_orig = d_1; // Accept reconstructed data
        Else:
            S_1 ← next combination of t + 1 shares;
            S_2 ← next combination of t + 1 shares with at least
            one differing share;
    Compare d_orig with desc from contract:
    If d_orig ≠ desc:
        F_sc.write(cid, "Challenge'', 1, S_1, S_2);
    Else: Proceed without dispute.
    /* Case 2: Detect and Verify Bad Shares
       After Reconstructing Data                                   */
    For each suspected bad share ds_{i,j}:
        Combine ds_{i,j} with t − 1 valid shares to reconstruct:
        d' ← F_ss.reconstruct(t, n, S');
        If d' ≠ d_orig:
            F_sc.write(cid, "Challenge'', 2, badshares);
On receive ("noComplain") from Self:
    F_sc.write(cid, "noComplain'', buyerID);
```

Dispute handling. $\mathcal{C}_{DEXO}$ provides a dispute-handling interface (the $Challenge$ function) to protect consumers from inconsistencies in data received from DEXO. This ensures robust consumer protection and enhances the fair exchange

process. We first define two key utilities used in verification:

- $\phi_1(t, n, shares[])$ verifies whether $shares[]$ can reconstruct the original data in the (t, n)-secret sharing scheme. $\phi_1()$ either returns a the reconstructed data d or returns an error code.
- $\phi_2(d, desc)$ verifies if the data d conforms to the format and normality description in $desc$ (e.g., whether d is within the numerical range). If verifies, $\phi_2()$ returns True; otherwise False.

$\phi_1()$ and $\phi_2()$ are implemented as $\mathcal{C}_{DEXO}$'s internal functions (implementation details are obviated in Algorithm 4).

Specifically, $\mathcal{C}_{DEXO}$ supports the following two dispute scenarios.

1) **Reconstructed data does not follow description:** If the consumer reconstructs the dataset $\{d_i\}_{i\in[M]}$ from received shares, but it does not match the format or attributes described in the contract, the consumer can invoke the contract's $challenge$ function. The contract first verifies share validity using Merkle Tree proofs, then applies ϕ_1 to reconstruct the data with two combinations of $t+1$ shares, ensuring consistency. It then uses ϕ_2 to validate the reconstructed data against the description. If the validation fails, the contract refunds all payments for the dataset.
2) **Bad shares from individual DEXO nodes:** If certain shares $\{ds_{i,j}\}$ from DEXO nodes are invalid and cannot reconstruct the data, the consumer invokes the $challenge$ function. The contract checks share integrity via Merkle Tree proofs and applies ϕ_1 to combine each bad share with $t-1$ valid shares. If the reconstruction fails, the contract flags the share as invalid and refunds payments to the corresponding node.

F. Optimization for On-chain Efficiency

The design so far requires the consumer server to establish at least t fair exchange sessions with the DEXO network with each session delivering one decryption key. To minimize the on-chain contract execution cost, we introduce two optimizations to allow the consumer to significantly reduce the number of contract calls while retrieving t data shares required for reconstructing the plaintext data.

Merged Query and Payment. Instead of having the consumer query each node (step ①) and make payments according to each exchange session (step ③) separately, they can batch-process the exchange sessions with all nodes in one contract call. That is, step ① is now a query on the shares from all N nodes; step ③ is now a payment for all queried shares.

Shared Key for $(t-F)$ Nodes. This method aims to minimize the number of fair exchange sessions without hampering the final delivery of requested data shares. Instead of having every node i generate a new share-encryption key k_i for its exchange with the consumer, the P-DApp server may pre-select $(t-F)$ nodes as the "priority group" with a certain node serving as the group leader (denoted $\mathcal{N}_p$) who will coordinate the generation of a common secret key k_p for the entire group (group key generation has been well studied and is thus orthogonal to our work). In this way, the consumer can invoke one exchange session with $\mathcal{N}_p$ to obtain the corresponding k_p for the data shares held by all $(t-F)$ nodes in this group (Step ② still

Algorithm 4 DEXO Contract $\mathcal{C}_{DEXO}$ Pseudocode

Data: dataSources, seller, buyer, price, $desc$, Δ, com, key, $keyRevealed$

```
/* Smart Contract Constructor                                   */
Function constructor(DEXO_Node_ID, price, desc,
 dataSources[], sellerNodes[]):
    Set seller ← DEXO_Node_ID
    Set price, desc, dataSources[], sellerNodes[]
/* Initialize data for sale                                     */
Function initialize( _Δ, _comm):
    Require sender in sellerNodes[] // sender: function caller
    Set Δ[sender] ← _Δ and commitment[sender] ← _comm
/* Buyer declares itself.                                       */
Function query():
    Set buyer.account← msg.sender
    Set buyer.desc = {_ip, _port} //specifying buyer's machine
     identifier
/* Buyer accepts z_i and transfers payment                      */
Function accept(buyer, sellerNode, payment):
    Require payment == price
    buyer deposit for sellerNode
/* Seller reveals the key                                       */
Function revealKey(key):
    Require sender in sellerNodes[]
    Require F_com.open(tid, commitment[sender]) == key)
    Set keyRevealed[sender] ← key
/* Optional dispute handling                                    */
Function challenge(casetype, shares1[], shares2[],
 badShares[], nodeID[], index[], proofs[]):
    Require sender == buyer
    // Verify all shares in MTHash Tree
    For j ← 0 to len(nodeID[]) do:
        k ← keyRevealed[sellerNode[j]]
        For each share in shares1[] ∪ shares2[] ∪ badShares[]:
            Require Δ[nodeID[j]] ==
              MTHash(Encrypt_k(share), proofs[i], index[i]);
    If casetype = 1: // Case 1: Validate Description
        d_1 ← φ_1(t, n, shares1[]);
        d_2 ← φ_1(t, n, shares2[]);
        Require d_1 == d_2; // Ensure consistency
        Require φ_2(d_1, desc); // Validate description
        If failed: Refund payments;
    If casetype = 2: // Case 2: Verify Bad Shares
        For j ← 0 to len(nodeID[]) do:
            For each badShare in badShares[]:
                d' ← φ_1(t, n, {badShare} ∪ shares1[: −1]);
                Require d' == d_1;
                If failed: Refund and mark invalid for badShare;
/* Buyer confirms no complaints                                 */
Function noComplain():
    Require sender == buyer
    Transfer funds to sellerNodes and data sources
Internal Function φ_1(t, n, S[]):
        d ← F_ss.reconstruct(t, n, S[]);// to instantiate on-chain
        Return d;
Internal Function φ_2(d, desc):
        Return True if d matches desc; False otherwise
```

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2025.3535671

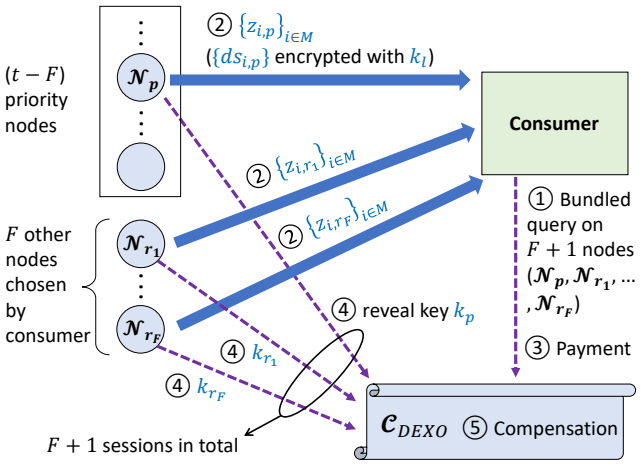


Fig. 2: Fair exchange process with merged query and shared key (as described in §V-E and §V-F). A total of $F+1$ parallel exchange protocol sessions are required.

need to be performed on each node). The consumer only needs to exchange with other F nodes to obtain the remaining F shares. In this way, the total number of fair exchange sessions has been reduced from t to $F+1$. If the system is configured $N \geq 3F+1$ and $t = \frac{2}{3}N$, this marks at least a 50% reduction in the sessions. We will show in Section VI-A that DEXO is still secure when this optimization is applied.

The overall procedure after the two optimizations are applied is shown in Fig. 2.

VI. Analyses

A. Security Analysis

In this section, we show that DEXO fulfills the proposed security goals under the defined threat model (Section III-C).

Theorem 1 (Data Source Verifiability): Each data source, i.e., a P-DApp user device, always performs data preprocessing, secret sharing, and signing as specified in the trusted application $\mathcal{F}_{TA}$ correctly. The execution integrity can be verified later by a DEXO node.

Proof: Establishing the trusted application $\mathcal{F}_{TA}$ as a TEE instance on a P-DApp user device ensures its execution integrity. The correctness of this process reduces to the integrity of TEE-based attested execution $\mathcal{G}_{att}$ as described in Section IV-B. Moreover, each signature within $\sigma_{i,j}$ ($j \in [N]$) generated by user i's $\mathcal{F}_{TA}$ instance proves the integrity of both the TEE runtime environment and the generated data share $ds_{i,j}$ which is verifiable by DEXO Nodes with the help of the TEE attestation service that verifies the validity of the user device's TEE public key. ■

This mechanism ensures that even if an adversary gains control over a P-DApp server or other parts of the system, they cannot tamper with the data processing or generate fraudulent data shares without detection. The TEE attestation process guarantees that only data shares produced by genuine, unaltered TEE instances are accepted by the DEXO nodes, effectively mitigating spoofing or data manipulation attacks.

Theorem 2 (End-to-end Confidentiality): The formatted data generated by a P-DApp user device i, d_i, is only delivered to the paid consumer while being hidden from other parties, including individual DEXO nodes and other consumers.

Proof: This property derives from the confidentiality of TEE-based attested execution $\mathcal{G}_{att}$ and the security of secrete sharing functionality $\mathcal{F}_{ss}$. More specifically, $\mathcal{G}_{att}$ generates d_i within the TEE and the (t, N)-secret sharing outputs only the data shares $\{ds_{i,j}\}$. Since we assume $F < t$, the compromised DEXO nodes cannot obtain enough data shares to reconstruct d_i. The same applies to malicious consumers who do not pay to receive decryption keys for at least t data shares of d_i. Therefore, DEXO prevents malicious nodes or non-paying consumers from reconstructing the original data the confidential data in the sale. ■

Theorem 3 (Fault Tolerance of Data Delivery): The consumer is guaranteed to receive the requested formatted data d_i despite the presence of up to F compromised DEXO nodes.

Proof: It is sufficient to show that the consumer can always receive the data shares to reconstruct the data. Since we assume $F < \frac{1}{2}N$, thus $N - F > F$. Therefore, there always exists a t so that $F < t \leq N - F$. This guarantees that the consumer can receive at least $N - F$ valid data shares from non-faulty nodes which can be used to reconstruct d_i (since $t \leq N - F$). ■

We remark that Theorem 2 and Theorem 3 imply the decentralization property since the system is resilient to single-point failure among DEXO nodes. This prevents any single or minority nodes from monopolizing or leaking the data.

Theorem 4 (Fair Exchange with Collusion Resistance): The P-DApp users receive compensation only if the consumer obtains the correct data. Simultaneously, the consumer receives the data only if the P-DApp users receive the requested compensation. This process is secure even if either party colludes with compromised DEXO nodes.

Proof: We first show the security against source-node collusion. Consider the worst-case scenario when F DEXO nodes are compromised and willing to collude with the P-DApp to scam a consumer without fulfilling the data provision. The F nodes can send crafted data shares to the consumer. Since the consumer receives at least t shares and $t > F$, it can always detect the inconsistencies of the received shares and invoke the dispute protocol (Section V-E) to abort the entire exchange and get the payment back. When the two optimizations (see §V-F) are applied, the consumer still gets at least $F + 1$ shares from each user; there is at least one correct share for the consumer to uncover the inconsistencies of the received shares. This provides sufficient evidence for the consumer to start a dispute process and claim the payment back (see §V-E).

Next, we show the security against consumer-node collusion. Consider the worst-case scenario when the F compromised DEXO nodes collude with the consumer to trick a P-DApp into providing the correct data without a full payment. The F nodes can send all their shares to the consumer, which however is not sufficient for the latter to reconstruct the original data since $F < t$. When the two optimizations (Section V-F) are applied, we consider the worst-case scenario when only one of the nodes with shared keys is compromised—this will make the data shares from all the $t - F$ nodes available

to the consumer. However, counting in the remaining $F-1$ compromised nodes, the consumer can still obtain at most $t-F+F-1=t-1$ shares, below the required t shares. Knowing the $t-1$ shares is no different from knowing one share since they both reveal no useful information about the original data.

Lastly, in case a consumer starts a dispute process, the same collusion resistance is achieved. This is shown in Lemma 1 below. ■

Lemma 1 (Integrity of Dispute Handling): The dispute handling process of DEXO is secure even if either party colludes with compromised DEXO nodes.

Proof: The security of the dispute-handling process relies on the integrity of the secret-sharing scheme and the predicate functions ϕ_1, ϕ_2. Even if a subset of DEXO nodes ($F < t$) colludes with one party (e.g., a consumer or a P-DApp), they cannot tamper with or fabricate valid data shares without detection. During a dispute, the contract first verifies the authenticity of all shares to ensure they originate from valid Merkle Tree roots. This step prevents a malicious consumer from injecting fake shares into the contract. After verifying the shares, the contract uses ϕ_1 to reconstruct the original data multiple times with different combinations of shares, ensuring consistency. It then applies ϕ_2 to validate that the reconstructed data matches the description specified in the contract. Any mismatch triggers a dispute and refunds payments. Furthermore, the contract checks each suspected bad share by combining it with $t-1$ valid shares and reconstructing the data using ϕ_1. If the reconstructed data does not match the verified original data, the share is flagged as invalid, and payments are refunded. The TEE attestation mechanism ensures that all data shares originate from trusted, unaltered TEE instances. Even if a consumer colludes with compromised nodes, they cannot generate sufficient valid shares to reconstruct the data without detection, as t valid shares are required. Similarly, a colluding P-DApp cannot cheat a consumer by providing incorrect data, as integrity checks during reconstruction will fail. Thus, DEXO's dispute handling guarantees fairness and integrity in resolving conflicts, even in the presence of malicious behaviors or collusion. ■

B. On-chain Complexity Analysis

Here we analyze how the number of P-DApp users M and the number of DEXO nodes N may affect the on-chain complexity. In our analysis, the optimal scenario assumes that all parties, including the buyer and DEXO nodes, act honestly according to the protocol. In this scenario, the transaction proceeds directly to completion without any disputes. In this optimal scenario, the P-DApp server constructs the contract and initializes the data exchange process. Each of the N DEXO nodes performs an Initialize operation to register metadata, such as Merkle root hashes, with the contract. The consumer submits a single query operation to request all necessary data shares, after which the N nodes verify the query and transmit encrypted data shares off-chain to the consumer. The consumer then makes t payments, corresponding to the t threshold shares required to reconstruct the original data. The N nodes verify payments, and t nodes detect and confirm receipt of payments. These t nodes proceed to execute the revealKey operation, releasing encryption keys that the consumer retrieves through t get key operations. Altogether, this process involves a total of $3N+3t+2$ contract calls, which are independent of the number of users (M) since the P-DApp server aggregates and packages data shares before interacting with the contract.

Intuitively, the irrelevance between M and the total number of contract calls is due to the fact that individual users do not interact with the contract by themselves. Instead, each DEXO node j collects the jth data share from all users and registers their cryptographic digest via one $initialize$ function call. The same applies to the disclosure of decryption keys, where node j makes one $revealKey$ function call to disclose k_j for decrypting all users' shares. When a fair exchange process finishes, the consumer invokes the contract via one $noComplain$ call to dispute payments to multiple users at once. These designs avoid the sheer amount of calls from data sources. Our system design reduces direct blockchain interactions. By delegating data storage and management to DEXO nodes, the design offloads resource-intensive processes, significantly reducing on-chain overhead.

However, the on-chain complexity also needs to take into account the blockchain execution cost (e.g., denominated in gas fee in Ethereum). In one fair exchange, the gas cost of calling $noComplain$ is dependent on the number of users M since the smart contract allocates payments to the users through M internal transactions. Even though an internal transaction requires minimal gas cost compared to a normal contract, the total gas consumption is still linear in M. We will demonstrate the gas cost of contract execution $noComplain$'s M-dependent cost in Section VIII-C. Specifically, the gas cost of this function scales linearly with M, requiring approximately 5735 units of gas for each additional user. This translates to an incremental cost of around \$0.22 per user under current Ethereum gas prices (at May 2024 Ethereum price). If a dispute happens, the gas cost of calling the $Challenge$ function is also dependent on M since the proof size submitted by a consumer may be proportional to the size of data shares received from one DEXO node.

Lastly, when the blockchain platform experiences congestion due to a high volume of transactions from external activities, DEXO consumers may still encounter increased transaction latency and elevated gas fees due to network competition, similar to all Web3 users in general. Addressing these fundamental limitations would require scalability solutions for the blockchain platform itself, such as integrating sidechains, payment channels or other Layer-2 mechanisms, or switching to private blockchains exclusive to our system. These considerations are of practical importance and we leave them to future study.

VII. Implementation

We provide a proof-of-concept implementation of DEXO's system components, including the TEE-capable P-DApp user,

DEXO node, data consumer, and the DEXO contract.[1]

We used Raspberry Pi 3 (RPi3) to simulate an IoT/mobile data provider (*i.e.*, a P-DApp user device) and OP-TEE [54] to implement the TEE-based attested execution functionalities based on RPi3's native ARM TrustZone support [21] and Open Portable TEE (OP-TEE) [54]. OP-TEE is an open-source implementation of the TEE concept primarily targeting ARM-based devices and is designed to provide a secure and isolated environment for running sensitive code and processing confidential data. The OP-TEE project can be compiled into the Linux system which can be run on the RPi3 board. We utilized the Repo Manifest [55] to compile and configure the various components involved. For the TrustZone TEE attestation, we utilized the attestation function of OP-TEE released in April 2022 [56]. For the secret-sharing-based data generation procedure, we ported an off-the-shelf implementation [57] of Shamir's Secret Sharing [50] into the TEE program $\mathcal{F}_{TA}$. The native RSA signature function is used to generate a signed attestation report at the end of $\mathcal{F}_{TA}$.

DEXO Contract. We implemented a proof-of-concept $\mathcal{C}_{DEXO}$, which is shown in Algorithm 4, with Solidity for the Ethereum blockchain containing about 130 lines of code. It realizes the important functions except the optional dispute handling routine described in Section V-E. The contract was deployed to the Ethereum Sepolia testnet for evaluating the gas and time cost of running the data exchanges.

VIII. Evaluation

We conducted experiments under varying conditions of DEXO to evaluate the following performance metrics: (i) Time and gas fee costs associated with using DEXO to obtain data as a data consumer, and its comparison to existing approaches. (ii) Time cost for a data provider to generate data in the TEE environment. The above metrics are evaluated for scalability under different number of data providers and DEXO nodes.

A. Time Cost of Transaction

Fig. 3 shows the time costs of invoking $\mathcal{C}_{DEXO}$ over the Sepolia testnet. It measures the average confirmation times (in milliseconds) for different smart contract functions: $initialize$, $buyerClaim$ (the equivalent of the DEXO contract's $query$ function), $accept$, and $revealKey$. Each test type is represented with a different color bar, and error bars indicate the standard deviation in the measurements.

For a buyer seeking to retrieve desired data after discovering it, the performance of the $buyerClaim$, $accept$, and $revealKey$ functions are particularly relevant. $buyerClaim$ and $accept$ have average confirmation times of approximately 12.5s. And $revealKey$ has average confirmation times of approximately 18s with a DEXO network with 20 nodes. This suggests that, under our test condition, a buyer can expect to complete these transactions and retrieve the data within 1 minute.

The extended confirmation times for the RevealKey function can be attributed to the complex validation procedures within the function. Specifically, the function includes checks to determine the validity of the provided key, which requires additional computational effort. This increased processing time on the testnet leads to longer intervals before a block can be confirmed. It is important to note that the computation results of a call are obtained before the block confirmation, not after.

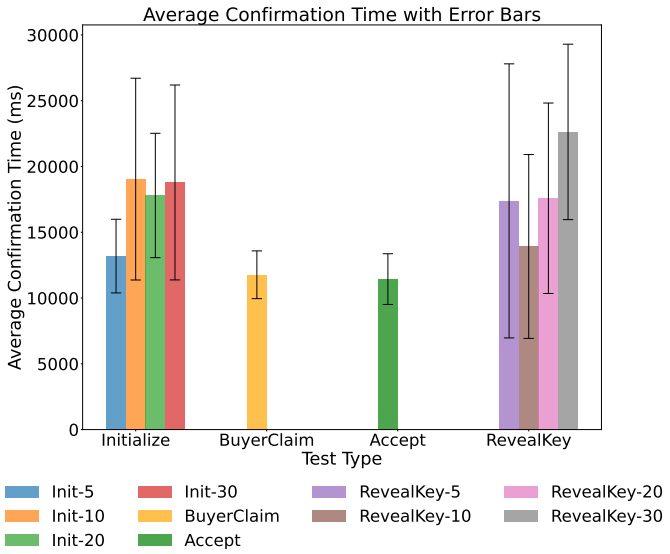


Fig. 3: Average Confirmation Times. Note: The suffix number represents the number of nodes used in the test. Those nodes call this function to the contract on the testnet at the same time

B. Gas Cost of DEXO

Execution of $\mathcal{C}_{DEXO}$ incur costs in terms of computational resources, which are quantified as gas costs in the Ethereum network. We evaluate the gas costs (on-chain execution fees) associated with various $\mathcal{C}_{DEXO}$ operations as shown in Table III. Firstly, deploying the DEXO smart contract is a *one-time* operation that sets up the contract on the Ethereum network. This computationally intensive operation results in gas consumption of approximately 2,325,998 units. In May 2024, with a gas price of 10.96 gwei and the value of Ether at $3,510 (USD), the approximate cost is about $89.48. However, it is essential to note that due to the volatile nature of Ether price and gas price on the Ethereum network, this cost can significantly fluctuate in real-world conditions.

The "initialize" function has a gas fee of 74,248 units ($2.85). The "noComplain" function is used for distributing the revenue. The base gas fee for invoking this function is 37,194 units ($1.43). In addition to this base fee, an extra 5,735 gas units ($0.22) are required for each data source included in the distribution. The gas fees for the "Accept", "revealKey", and "check Key" functions in Solidity are fixed at 74,843, 84,334, and 3,457 units (i.e., $2.87, $3.24, and $0.13), respectively, and do not vary with the number of elements in the dataSources array.

[1]Our code is available at https://github.com/yli568/DEXO.

TABLE III: Gas Costs of Invoking $\mathcal{C}_{DEXO}$ Functions

Function	gas fee (Units) & Cost (USD)
Deployment	2,325,998 ($89.48)
Initialize	74,248($2.85)
noComplain	37,194 ($2.05) + 5,735 ($0.22) × #DSs
Accept	74,843 ($2.87)
revealKey	84,334 ($3.24)
check Key	3,457 ($0.13)

C. Comparing Gas Cost of DEXO with Chainlink

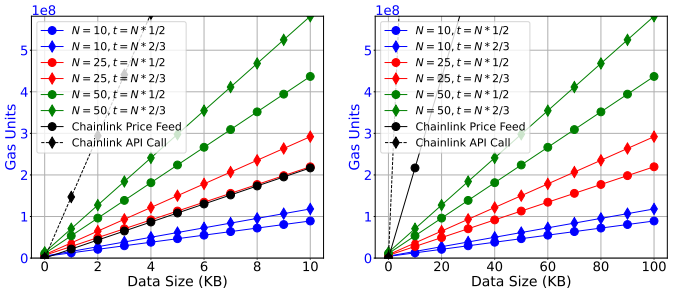
We compare the on-chain gas cost with the popular data oracle solution Chainlink [17], which delivers data through contract API calls. According to [58], the gas fee for a "Price Feed" transaction is 216,844 units ($8.34), while an API Call incurs a gas fee of 1,470,295 units ($56.56). A single call through Price Feeds or an API Call on Chainlink or its Oracles typically retrieves a singular data point, such as the current weather condition at a specific time, the real-time price of a cryptocurrency, or the current market value of a specific stock. In comparison, DEXO delivers the ciphertext data off-chain and only uses blockchain for fair exchange transactions, significantly reducing the on-chain gas cost.

Based on the above benchmarking result, we provide an extrapolation analysis of on-chain gas costs when we scale up the number of data providers and compare them to Chainlink The results are shown in Fig. 4(a) and Fig. 4(b). As the number of DEXO nodes increases, the gas fee increases linearly because the buyer must transact with more nodes to acquire a sufficient number of shares for aggregating usable data. When $t = \frac{2}{3}n$, the gas fee incurred is higher than that when $t = \frac{1}{2}n$ because a smaller threshold signifies the need for fewer shares to aggregate usable data, thereby implying fewer transactions.

Specifically, Fig. 4(a) represents the scenario where each P-DApp user contributes 10 bytes of data per instance. An increase in Data Size implies an increase in the number of users. For the two methods of Chainlink, an increase in Data Size signifies a rise in the number of requests made to Chainlink. The result shows that if each P-DApp user contributes a small amount of data per instance, the gas fee for Chainlink Price Feed is roughly equivalent to the gas fee for DEXO when $n = 25$ and $t = \frac{1}{2}n$. Moreover, all the scenarios for DEXO listed outperform the Chainlink API Call regarding gas fees. Fig. 4(b) depicts the scenario when each P-DApp user provides 100 bytes of data per instance. This figure illustrates that if each P-DApp user contributes slightly larger amounts of data per instance, all scenarios for DEXO outperform the Chainlink Price Feed. This further highlights DEXO's significant advantage in transmitting larger volumes of data from an increasing number of data providers.

D. TEE Overhead

We evaluated the computation overhead attributed by the ARM TrustZone TEE on the P-DApp user end. We tested the TEE program $\mathcal{F}_{TA}$ to generate data, shares, and sign a single share with the runtime environment. Based on the



(a) (b)

Fig. 4: Compare Gas Cost of DEXO with Chainlink. (a) Each user provides each share of a one-time 10B data to each Node. (b) Each user provides each share of a one-time 100B data to each Node.

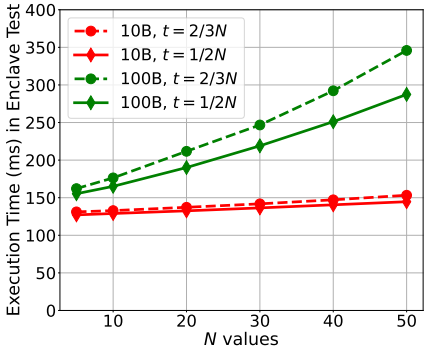


Fig. 5: Time cost of data provider's TEE operation ($\mathcal{F}_{TA}$). Here '10B' refers to the size of the original data involved in the secret sharing process, which is 10 bytes. Similarly, '100B' indicates that the original data size is 100 bytes.

testing conducted with OP-TEE on RPi3, it has been observed that generating shares in TEE and signing them with the TA execution environment costs variant based on the size of the original data and the number of shares that DEXO needs, which is the number of nodes in the DEXO network.

Fig. 5 shows the execution time (ms) in an enclave test for varying node counts (N) in a DEXO network, with N values ranging from 10 to 50. The y-axis ranges from 0 to 400 ms. For 10B origin data (red lines), we observe that (i) the execution time increases relatively slowly across all N values and (ii) the difference between thresholds $t = N/2$ (solid) and $t = 2N/3$ (dashed) is not significant. For 100B origin data (green lines), we observe that (i) the execution time increases with the number of nodes faster and (ii) the threshold $t = N/2$ (solid) is consistently lower than $t = 2N/3$ (dashed).

Overall, execution time increases with larger data sizes and higher threshold values, especially as the number of nodes grows. In practice, we can assume a fixed number of N, similar to existing DON solutions (e.g., Chainlink), and a maximum size of data entry. In future work, we will explore more efficient TEE-based secret-sharing implementations to reduce the data provider's operational cost further.

IX. Conclusion

In this work, we introduced a new decentralized data exchange mechanism called DEXO for enabling a secure and scalable marketplace for IoT and mobile data. By augmenting the decentralized oracle network paradigm with innovative hardware-cryptographic co-design that harmonizes trusted hardware, secret sharing, and blockchain smart contract, DEXO for the first time enables secure data exchange between distributed data providers and consumers while fulfilling end-to-end data confidentiality, source verifiability, decentralization, and fairness goals with strong resilience to participant failures and collusions. The experiment results demonstrate DEXO's feasibility in the deployment with Ethereum smart contracts with moderate on-chain gas cost per unit of data consumed while incurring minimal off-chain execution overhead for individual data providers.

Acknowledgments

This work was supported in part by the US National Science Foundation under grant numbers 2247561 and 2238680, and by the Office of Naval Research under grant number N00014-24-1-2730 subawarded through Virginia Tech.

References

[1] FCC, "FCC Proposes $91.6M Fine against T-Mobile in Location Information Case." https://www.fcc.gov/document/fcc-proposes-916m-fine-against-t-mobile-location-information-case, 2020.

[2] FCC, "FCC Proposes $57.2M Fine against AT&T in Location Information Case." https://www.fcc.gov/document/fcc-proposes-572m-fine-against-att-location-information-case, 2020.

[3] FCC, "FCC Proposes $48.3M Fine against Verizon in Location Information Case." https://www.fcc.gov/document/fcc-proposes-483m-fine-against-verizon-location-information-case, 2020.

[4] J. Sherman, "Data brokers and data breaches," *Duke University's Sanford School of Public Policy*, 2022.

[5] S. W. Driessen, G. Monsieur, and W.-J. Van Den Heuvel, "Data market design: a systematic literature review," *Ieee access*, vol. 10, pp. 33123–33153, 2022.

[6] P. Missier, S. Bajoudah, A. Capossele, A. Gaglione, and M. Nati, "Mind my value: a decentralized infrastructure for fair and trusted iot data trading," in *Proceedings of the Seventh International Conference on the Internet of Things*, pp. 1–8, 2017.

[7] A. Suliman, Z. Husain, M. Abououf, M. Alblooshi, and K. Salah, "Monetization of iot data using smart contracts. iet netw 8: 32–37," 2018.

[8] S. Bajoudah, C. Dong, and P. Missier, "Toward a decentralized, trustless marketplace for brokered iot data trading using blockchain," in *2019 IEEE international conference on blockchain (Blockchain)*, pp. 339–346, IEEE, 2019.

[9] Databroker, "Databroker official website." https://www.databroker.global/, 2020.

[10] O. Protocol, "A decentralized data exchange protocol to unlock data for ai," 2020.

[11] G. S. Ramachandran, R. Radhakrishnan, and B. Krishnamachari, "Towards a decentralized data marketplace for smart cities," in *2018 IEEE International Smart Cities Conference (ISC2)*, pp. 1–8, IEEE, 2018.

[12] P. Sharma, S. Lawrenz, and A. Rausch, "Towards trustworthy and independent data marketplaces," in *Proceedings of the 2020 The 2nd International Conference on Blockchain Technology*, pp. 39–45, 2020.

[13] Rarible, "Rarible official website." https://rarible.com/, 2020.

[14] OpenSea, "Opensea official website." https://opensea.io/, 11 2017.

[15] Chainlink, "What is the blockchain oracle problem?." https://blog.chain.link/what-is-the-blockchain-oracle-problem/, 2020. [Accessed: 1/5/2023].

[16] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *Proceedings of the 2016 aCM sIGSAC conference on computer and communications security*, pp. 270–282, 2016.

[17] L. Breidenbach, C. Cachin, B. Chan, A. Coventry, S. Ellis, A. Juels, F. Koushanfar, A. Miller, B. Magauran, D. Moroz, *et al.*, "Chainlink 2.0: Next steps in the evolution of decentralized oracle networks," 2021.

[18] Band Protocol, "Bandchain whitepaper." https://docs.bandchain.org/whitepaper/, 2022.

[19] B. Benligiray, S. Milic, and H. Vänttinen, "Decentralized apis for web 3.0," *API3 Foundation Whitepaper*, 2020.

[20] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A decentralized truth discovery approach to the blockchain oracle problem," in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*, IEEE, 2023.

[21] ARM, "Arm trustzone technology." https://developer.arm.com/ip-products/security-ip/trustzone, 2023. [Accessed: 1/5/2023].

[22] Apple, "Apple t2 secure chip," 2019. [Accessed: 1/5/2023].

[23] "Op-tee: attestation.c." https://github.com/OP-TEE/optee_os/blob/master/core/pta/attestation.c, 3 2023.

[24] L. Eckey, S. Faust, and B. Schlosser, "Optiswap: Fast optimistic fair exchange," in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, pp. 543–557, 2020.

[25] S. Dziembowski, L. Eckey, and S. Faust, "Fairswap: How to fairly exchange digital goods," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 967–984, 2018.

[26] Y. Xiao, N. Zhang, J. Li, W. Lou, and Y. T. Hou, "Privacyguard: Enforcing private data usage control with blockchain and attested off-chain contract execution," in *European Symposium on Research in Computer Security*, pp. 610–629, Springer, 2020.

[27] F. Zhang, D. Maram, H. Malvai, S. Goldfeder, and A. Juels, "Deco: Liberating web data using decentralized oracles for tls," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1919–1938, 2020.

[28] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.

[29] "EOS.IO technical white paper v2." https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md, 2018.

[30] H. Papacharissiou, "How to build a dapp in three steps." https://blog.chain.link/how-to-build-a-dapp, 2023. [Accessed: 1/5/2023].

[31] Streamr, "Streamr: Decentralized real-time data economy. white paper.." https://streamr.network/, 2017.

[32] H. Pagnia, F. C. Gärtner, *et al.*, "On the impossibility of fair exchange without a trusted third party," tech. rep., Citeseer, 1999.

[33] A. Alsharif and M. Nabil, "A blockchain-based medical data marketplace with trustless fair exchange and access control," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pp. 1–6, IEEE, 2020.

[34] D. Lopez and B. Farooq, "A multi-layered blockchain framework for smart mobility data-markets," *Transportation Research Part C: Emerging Technologies*, vol. 111, pp. 588–615, 2020.

[35] M. Campanelli, R. Gennaro, S. Goldfeder, and L. Nizzardo, "Zero-knowledge contingent payments revisited: Attacks and payments for services," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 229–243, 2017.

[36] M. Herlihy, "Atomic cross-chain swaps," in *Proceedings of the 2018 ACM symposium on principles of distributed computing*, pp. 245–254, 2018.

[37] I. Tsabary, M. Yechieli, A. Manuskin, and I. Eyal, "Mad-htlc: because htlc is crazy-cheap to attack," in *2021 IEEE symposium on security and privacy (SP)*, pp. 1230–1248, IEEE, 2021.

[38] S. Wadhwa, J. Stoeter, F. Zhang, and K. Nayak, "He-htlc: Revisiting incentives in htlc," *Cryptology ePrint Archive*, 2022.

[39] J. Guarnizo and P. Szalachowski, "Pdfs: practical data feed service for smart contracts," in *European Symposium on Research in Computer Security*, pp. 767–789, Springer, 2019.

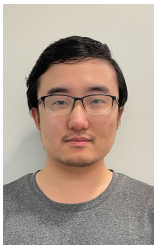
[40] B. Benligiray, S. Milic, and H. Vänttinen, "Decentralized APIs for Web 3.0," *API3 Foundation Whitepaper*, 2020.

[41] WINkLink, "Introduction to WINkLink." https://doc.winklink.org/v1/doc/en/, 2022. [Accessed: 1/5/2023].

[42] "The 3 levels of data aggregation in chainlink price feeds." https://blog.chain.link/levels-of-data-aggregation-in-chainlink-price-feeds/, 2023. [Accessed: 5/31/2023].

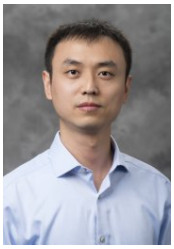
[43] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller, and D. Song, "Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 185–200, IEEE, 2019.

[44] V. Costan and S. Devadas, "Intel sgx explained," *Cryptology ePrint Archive*, 2016.
[45] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution.," in *HASP@ISCA*, p. 10, 2013. [Accessed: 2/1/2023].
[46] AMD, "Secure encrypted virtualization (sev)." https://developer.amd.com/sev/, 2020. [Accessed: 1/5/2023].
[47] "Intel® software guard extensions: Strengthen enclave trust with attestation." https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/attestation-services.html, 1 2023.
[48] "Integritee network docs." https://docs.integritee.network/, 2021.
[49] R. Pass, E. Shi, and F. Tramer, "Formal abstractions for attested execution secure processors," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 260–289, Springer, 2017.
[50] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.
[51] O. Goldreich, *Foundations of Cryptography*. Cambridge University Press, 2001.
[52] R. Canetti and M. Fischlin, "Universally composable commitments," in *Advances in Cryptology—CRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–23, 2001 Proceedings 21*, pp. 19–40, Springer, 2001.
[53] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pp. 136–145, IEEE, 2001.
[54] Trusted Firmware Project, "Op-tee on github." https://github.com/OP-TEE, 2017.
[55] "Repo manifest for op-tee development." https://github.com/OP-TEE/manifest/, 2023.
[56] D. Harbin, "Trusted firmware op tee: v3.17.0 release." https://www.trustedfirmware.org/blog/Trusted-Firmware-OPTEE-Release-3-17-0/, 4 2022.
[57] fletcher, "c-sss: A c implementation of shamir's secret sharing," 2020.
[58] K. M. Khan, R. Taufique, and M. A. Rauf, "Investigation on a price oracle problem," *Mehran University Research Journal of Engineering and Technology*, vol. 41, no. 4, pp. 138–145, 2022.

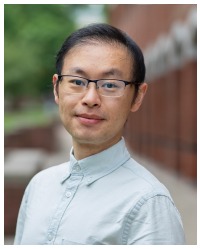


Yue Li (Student Member, IEEE) received the M.Sc. degree in Computer Science from the Department of Computer Science, University of Kentucky, Lexington, KY, USA, in 2023 and is currently a Computer Science Ph.D. student in the same department. He received his bachelor's degree in Software Engineering from the University of Electronic Science and Technology of China, Chengdu, Sichuan, China. His research interests lie in distributed system security and decentralized systems.

Ifteher Alom (Student Member, IEEE) is a Computer Science graduate student at the University of Kentucky, Lexington, KY, USA. He received his bachelor's degree from the Department of Computer Science, Shahjalal University of Science and Technology, Sylhet, Bangladesh. His research interests include applied cryptography, identity management, and blockchain applications.



Wenhai Sun (Senior Member, IEEE) holds two Ph.D. degrees from the School of Telecommunications, Xidian University, and from the Department of Computer Science, Virginia Tech, respectively. He is currently an Assistant Professor with the Department of Computer and Information Technology, Purdue University, West Lafayette, IN, USA, and a Faculty Member affiliated with the Center for Education and Research in Information Assurance and Security at Purdue. His research interests lie in privacy-enhancing technologies, confidential computing, AI/ML security, decentralized trust and infrastructure, and cyber-physical systems security. Dr. Sun won the Distinguished Paper Award in ACM ASIACCS 2013. He received the NSF CAREER Award in 2023.



Yang Xiao (Member, IEEE) received the Ph.D. degree in Computer Engineering from the Department of Electrical and Computer Engineering, Virginia Tech, USA in 2022. He is currently an Assistant Professor with the Department of Computer Science at the University of Kentucky, Lexington, KY, USA. His research interests lie in network security, distributed system security, blockchain and decentralized systems, and mobile network security.