Towards Probabilistic Semantic Assurance in Decentralized Data Exchange

Christos Karapapas[1,2], Iakovos Pittaras[2], George C. Polyzos[3], and Constantinos Patsakis[1,4]

[1]Athena Research Center, Greece
[2]Department of Informatics, Athens University of Economics and Business, Greece
[3]School of Data Science, The Chinese University of Hong Kong, Shenzhen, China
[4]University of Piraeus, Greece

Abstract

Decentralized data markets require fair exchange between providers, who do not want to reveal plaintext data before payment, and requesters, who need evidence that the data is useful before paying. Existing blockchain-based fair-exchange protocols provide strong guarantees for atomicity and delivery correctness, but they do not directly address semantic uncertainty: a correctly delivered dataset may still be irrelevant, corrupted, mislabeled, or inconsistent with its advertised description.

We propose a lightweight fair data exchange protocol that combines Zero-Knowledge Proofs (ZKPs) of key possession with cut-and-choose semantic sampling. A provider stores encrypted chunks off-chain and registers an encrypted manifest reference together with public keys, key commitments, and a ZKP of key possession. The requester escrows the payment and challenges a random subset of chunks. The provider reveals the sampled keys, allowing local plaintext inspection before final approval. The protocol provides key validity, probabilistic key applicability, and probabilistic semantic assurance, without claiming absolute semantic fairness.

Keywords: Fair exchange; Semantic assurance; Blockchain; Zero Knowledge Proofs (ZKPs)

1 Introduction

Data is increasingly exchanged through online marketplaces, decentralized storage systems, and blockchain-mediated platforms. This trend appears across several domains. In Artificial Intelligence, model developers acquire training, fine-tuning, evaluation, and annotation datasets from external providers, but often may assess quality only after gaining access. In healthcare and biomedical research, hospitals, laboratories, and research consortia exchange de-identified clinical records, imaging datasets, genomic data, or cohort-level statistics under strict access constraints. In finance, firms purchase alternative data such as transaction aggregates, market signals, credit-risk indicators, and web-derived intelligence, where freshness and semantic relevance directly affect economic value.

However, fair exchange remains a core challenge in data exchange processes. A data provider may be unwilling to disclose valuable plaintext data before payment, while a data requester may be unwilling to pay until they have evidence that the data is usable, complete, and consistent with its advertised description. This creates a trust gap that conventional escrow mechanisms do not fully address.

Smart contracts can enforce deterministic payment logic, but they cannot directly evaluate whether off-chain data is semantically meaningful. For example, a smart contract can verify

that a key matches a public commitment, but it cannot determine whether decrypted records correspond to a promised dataset, whether the records are corrupted, or whether the data quality satisfies the requester's expectations. This mismatch between on-chain verifiability and off-chain semantic value motivates the need for hybrid protocols that combine cryptographic enforcement with requester-side inspection.

This paper presents a lightweight fair data exchange protocol for decentralized data markets. The central idea is to separate three notions of assurance. First, key validity ensures that the provider knows the secret keys corresponding to previously committed public keys. Second, key applicability ensures that these keys decrypt the encrypted chunks offered for sale. Third, semantic assurance refers to the protocol's ability to enable requester-side inspection of randomly challenged plaintext chunks before payment. Rather than automatically evaluating semantic properties, the protocol enables the requester to evaluate them by inspecting an authenticated random sample before settlement, thereby providing probabilistic confidence that the advertised dataset is suitable for its intended use. The first property can be addressed cryptographically using zero-knowledge proofs of key possession. The second and third properties are addressed via a cut-and-choose sampling mechanism, in which the requester verifies a randomly selected subset of chunks before authorizing the final settlement.

Existing blockchain-based fair-exchange protocols provide strong guarantees of atomicity and delivery correctness, but they typically focus on whether the delivered digital object matches a commitment, a predicate, or a decryption condition. These guarantees are necessary but not sufficient for data markets in which the value of the exchanged object is semantic- and application-dependent. A dataset may be correctly encrypted, correctly committed, and correctly delivered, while still being irrelevant, outdated, corrupted, mislabeled, incomplete, or otherwise unsuitable for the requester's intended use. This motivates a complementary notion of semantic assurance; before final settlement, the requester should obtain evidence about the plaintext content itself, without requiring the provider to reveal the entire dataset before payment.

The contribution of this paper is threefold. First, we formulate a direct provider–requester fair data exchange setting in which the central challenge is not only atomic payment, but also semantic uncertainty about the purchased dataset. Second, we propose a chunk-based protocol that combines IPFS [3] content addressing, manifest-level binding, and Zero-Knowledge Proofs (ZKPs) of key possession to cryptographically link encrypted chunks with their corresponding decryption material. Third, we introduce a cut-and-choose semantic inspection phase that gives the requester probabilistic assurance about dataset quality before final settlement. Since semantic quality cannot generally be encoded as an on-chain predicate, our protocol does not claim absolute semantic fairness; rather, it provides a trust-minimized mechanism for combining cryptographic enforceability with requester-side plaintext inspection.

2 Protocol Workflow

The proposed Decentralized Fair Data Exchange (DFE) protocol facilitates a trust-minimized exchange between a Data Provider (DP) and a Data Requester (DR) through a series of on-chain and off-chain interactions, providing cryptographic binding, key-delivery accountability, and probabilistic semantic assurance. We assume that the requester has a public encryption key pk_{DR}, used to receive encrypted protocol references.

Step 0: Provider preprocessing. The data provider DP partitions the dataset into N chunks $M_1, \ldots, M_N$. For each chunk i, the provider samples a symmetric encryption key K_i and an asymmetric Elliptic Curve Integrated Encryption Scheme (ECIES) key pair (sk_i, pk_i). The provider encrypts the data chunk under the symmetric key, $C_i = Enc_{K_i}^{AES}(M_i)$, and wraps the symmetric key using ECIES under the corresponding public key, $W_i = Enc_{pk_i}^{ECIES}(K_i)$. The provider also computes a commitment to the ECIES private key using Poseidon, a cryptographic

hash function designed for efficient use in zero-knowledge proof systems: $com_i = Poseidon(sk_i)$.

Step 1: Manifest construction and private manifest-reference encryption. The provider uploads the encrypted chunks C_i to the off-chain storage layer and obtains their object references CID_i. The provider then constructs a manifest $\mathcal{M} = \{(i, CID_i, W_i, pk_i, com_i)\}_{i=1}^{N}$, which binds each chunk index to its encrypted object reference, wrapped symmetric key, ECIES public key, and key commitment.

The manifest is uploaded to the off-chain storage layer, producing a content-addressed reference $CID_{\mathcal{M}}$. Instead of publishing this reference in plaintext, the provider encrypts it under the requester's public encryption key: $E_{CID} = Enc^{hyb}_{pk_{DR}}(CID_{\mathcal{M}})$. The encrypted reference E_{CID} is later registered on-chain, allowing only the requester to recover the manifest location.

Step 2: Commitment and proof registration. The provider submits to the smart contract the encrypted manifest reference E_{CID}, the ECIES public keys $\{pk_i\}_{i=1}^{N}$, the key commitments $\{com_i\}_{i=1}^{N}$, and a batch zero-knowledge proof of key possession π_{pos}.

The proof establishes that the provider knows private keys $\{sk_i\}_{i=1}^{N}$ such that, for every registered index i, $pk_i = sk_i \cdot G$ and $com_i = Poseidon(sk_i)$. This step binds the provider to the registered public keys and commitments before challenge generation, while keeping the manifest location hidden from public observers.

Step 3: Manifest retrieval and escrow. The requester retrieves E_{CID} from the smart contract and decrypts it using their private key: $CID_{\mathcal{M}} = Dec_{sk_{DR}}(E_{CID})$. The requester then uses $CID_{\mathcal{M}}$ to retrieve the manifest $\mathcal{M}$ from the off-chain storage layer. From the manifest, the requester obtains the encrypted chunk references CID_i, the wrapped symmetric keys W_i, and the corresponding public verification data.

Before depositing escrow, the requester checks that the manifest is retrievable and that the referenced encrypted chunks are available. The requester then deposits the purchase price into the smart contract escrow, thereby initiating the exchange. The requester also checks that the pk_i and com_i values in the retrieved manifest match the values registered in the smart contract.

Step 4: Requester-seeded random challenge generation. Once E_{CID}, the registered public keys, commitments, and escrow are fixed, the requester provides a previously committed random seed r_{DR}. The smart contract derives the challenge subset as $K = Sample(H(r_{DR}, E_{CID}, sessionId), N$ where $Sample(\cdot)$ deterministically returns k distinct indices from $\{1, \ldots, N\}$. Thus, the requester contributes randomness but does not manually select the challenged chunks, while the provider cannot adapt the committed manifest or key material after the challenge is derived.

Step 5: Sample-key reveal. For every challenged index $i \in K$, the provider reveals the ECIES private key sk_i. The requester verifies that the revealed key matches the registered public key and commitment: $pk_i = sk_i \cdot G$ and $com_i = Poseidon(sk_i)$. Only keys that pass these checks are used for sample decryption.

Step 6: Sample decryption and semantic inspection. For every challenged index $i \in K$, the requester obtains the corresponding CID_i and W_i from the privately retrieved manifest. The requester uses the revealed ECIES private key sk_i to recover the wrapped symmetric key $K_i = Dec^{ECIES}_{sk_i}(W_i)$, and decrypts the corresponding encrypted chunk $M_i = Dec^{AES}_{K_i}(C_i)$. The requester then inspects the recovered plaintext chunks locally, checking whether they match the advertised dataset description, schema, format, and expected content. This provides probabilistic semantic assurance, since only a random subset of chunks is inspected.

Step 7: Approval or recovery branch. If the sampled plaintexts are acceptable, the requester submits approval to the smart contract, enabling final key delivery and settlement. If the sample is rejected, or the requester remains silent until timeout, the protocol enters a recovery branch according to the contract policy. This protects the requester against detected invalid samples, but it does not fully prevent requester-side griefing, since semantic acceptance remains a client-side judgment.

Step 8: Final key reveal. After requester approval, the provider reveals the ECIES private keys for the remaining, unchallenged indices $U = \{1, \ldots, N\} \setminus K$. For every $i \in U$, the

smart contract verifies that the revealed key matches the registered public key and commitment: $pk_i = sk_i \cdot G$ and $com_i = Poseidon(sk_i)$.

Step 9: Automated escrow settlement. If all required final keys pass the on-chain verification checks, the smart contract releases the escrowed payment to the provider. If the provider fails to reveal valid keys before the delivery deadline, the requester is refunded according to the contract policy.

The requester then uses the revealed keys together with the privately retrieved manifest to unwrap the corresponding symmetric keys and decrypt the full dataset. Since the manifest reference is encrypted under the requester's public key, public observers cannot reconstruct the dataset from blockchain data alone. However, if the manifest reference leaks off-chain, publicly revealed final keys may allow third parties to decrypt the corresponding chunks.

3 Informal Security Analysis and Limitations

The proposed protocol is designed to provide trust-minimized fair data exchange by combining three complementary mechanisms: IPFS-based manifest binding, zero-knowledge key-possession proofs, and cut-and-choose plaintext inspection. We provide an informal security analysis that identifies the main guarantees and limitations of the design.

Fixed dataset population The random challenge is meaningful only if it is sampled from a dataset population that has been fixed before the challenge is generated. To achieve this, the provider stores encrypted chunks off-chain and constructs a manifest containing tuples of the form $(i, CID_i, W_i, pk_i, com_i)$, where i is the chunk index, CID_i is the object reference of the encrypted chunk, W_i is the ECIES-wrapped symmetric key, pk_i is the corresponding ECIES public key, and com_i is the commitment to the associated private key. The manifest is uploaded to the off-chain storage layer, and its content-addressed reference is encrypted under the requester's public key as $E_{CID} = Enc^{hyb}_{pk_{DR}}(CID_{\mathcal{M}})$. The encrypted reference, together with the public keys and commitments, is registered on-chain before challenge generation. Since the manifest reference and registered key material are fixed before sampling, the provider cannot adapt the sampled population after observing the challenge.

Key validity The zero-knowledge proof layer establishes that the provider knows the private keys corresponding to the public keys and commitments registered in the manifest. For each chunk i, the intended relation proves knowledge of sk_i such that $pk_i = sk_i \cdot G$ and $com_i = Poseidon(sk_i)$. This prevents the provider from registering arbitrary public keys or commitments without knowing the corresponding secret keys. Importantly, the proof establishes key possession, not data correctness. It does not prove that the encrypted chunk is meaningful, relevant, or consistent with the advertised dataset description.

ZK setup assumptions The circuit is assumed to be public, open source, and fixed before the exchange, with its verification key bound to the deployed smart contract. PLONK is a suitable candidate for this layer because it supports a universal, updatable setup, allowing the verification key to be generated from public ceremony parameters rather than from a provider-specific trusted setup.

Key applicability semantic assurance After the requester escrows the payment, a random subset $K \subseteq \{1, \ldots, N\}$ is selected. The provider reveals the secret keys for the challenged chunks, and the requester decrypts the corresponding plaintexts locally. This step provides evidence that the revealed keys apply to the encrypted chunks and allows the requester to assess whether the sampled plaintexts are consistent with the advertised description. The protocol

does not automatically evaluate semantic properties or determine whether the dataset satisfies application-specific requirements. Instead, it enables the requester to perform this assessment through authenticated inspection of randomly selected plaintext chunks before authorizing settlement. The guarantee is probabilistic: if the provider includes m bad chunks among N total chunks and the requester challenges k chunks uniformly at random without replacement, the probability of detecting at least one bad chunk is $P_{\mathsf{detect}} = 1 - \frac{\binom{N-m}{k}}{\binom{N}{k}}$. Thus, larger challenge sets increase semantic assurance but also reveal more information before final settlement.

Atomic settlement If the requester accepts the sampled plaintexts, the protocol proceeds to final settlement. In the baseline design, the provider reveals the remaining ECIES private keys on-chain. The smart contract verifies that each revealed key matches the registered public key and commitment, and releases the escrowed payment only if all required checks pass. This provides contract-verifiable key delivery and automated settlement.

Limitations The protocol does not provide absolute semantic fairness. A provider may still include bad chunks that are not selected during the challenge phase. The effectiveness of semantic sampling also depends on how the dataset is partitioned into chunks; strategic chunk construction could reduce sample representativeness. Standardized or deterministic chunking policies may mitigate this issue. The protocol also does not eliminate requester griefing: a malicious requester, possibly controlling multiple identities or colluding with other requesters, may repeatedly initiate exchanges, inspect different challenged chunks, and refuse to approve final settlement, gradually learning an increasing fraction of the dataset without payment. Possible mitigations include non-refundable sampling fees, requester bonds, reputation mechanisms, or arbitration. Finally, IPFS does not by itself guarantee availability; practical deployments may require pinning, replication, or storage-incentive mechanisms [8].

4 Related Work

Fair exchange has been extensively studied as a mechanism for enabling mutually distrusting parties to exchange digital assets without giving either party an unfair advantage. Early protocols relied on trusted third parties to mediate the exchange and resolve disputes [2, 1]. Blockchain-based protocols replace this mediator with smart contracts, which can hold escrowed assets, enforce public verification rules, and execute settlement logic in a decentralized manner [13]. This shift has motivated a broad line of decentralized fair exchange protocols with different storage models, trust assumptions, and dispute mechanisms. Zeng et al. [14] provide a comprehensive overview of such protocols and their security guarantees.

Several blockchain-based schemes focus on ensuring atomic exchange between data and payment. FairSwap [7] employs cryptographic commitments and dispute resolution mechanisms to guarantee that either both parties obtain the expected outcome or neither party does. Li et al. [9] propose Fairtrade, a decentralized framework for fair online data trading. The authors propose two solutions, one utilizes homomorphic encryption and data sampling techniques, while the second uses double-authentication-preventing signatures with smart contracts. A more recent work by Tas et al. [12] proposes a blockchain Fair Data Exchange (FDE) protocol that enables a storage server to atomically transfer a data file to a client; the client receives the file if and only if the server receives an agreed-upon payment. To achieve this, the authors rely on blockchain mechanisms to enforce the atomicity of the exchange and use verifiable encryption under a committed key (VECK) to ensure that the client receives the correct data before releasing the payment for the decryption key. These works provide strong guarantees for atomicity and delivery correctness. However, they do not explicitly address semantic fairness from the buyer's perspective: even if the delivered data matches a commitment, the commitment

alone does not guarantee that the data is relevant, non-garbage, or consistent with the advertised description.

To provide stronger guarantees regarding data validity, several studies combine blockchain technologies with sampling-based verification techniques. In particular, Chenli et al. [4] introduce a fair trading platform that guarantees both exchange fairness and revenue-distribution fairness. Regarding exchange fairness, the authors propose a new key verification mechanism that avoids relying on generic zk-SNARK-based proofs for data correctness verification, and present a novel atomic exchange protocol based on Hashed Timelock Contracts on Ethereum. Regarding distribution fairness, they propose a verifiable statement protocol that efficiently splits income between the broker and sellers. The protocol aims to guarantee the validity of the encrypted data and the keys sent by a random sampling algorithm. The data is divided into n slices. For each slice d_i, the broker uses a different key k_i to encrypt it using AES. Then, the broker generates a public/private key pair for each k_i and encrypts it. Then, a set of keys will be randomly selected and sent to the buyer, who verifies them. After the data correctness verification, the atomic exchange happens. This step is based on private key lock transactions [6]. Fair2Trade is the closest related work to our overall protocol design, since it allows buyers to inspect plaintext slices before the final exchange. However, Fair2Trade is blockchain-based but broker-mediated: sellers assign data to a broker, who sells it to buyers and later distributes revenue. Our work instead considers a direct provider–requester exchange without a broker or revenue-distribution layer. Moreover, while the protocol builds on earlier sampling-based fair exchange techniques, such as the cut-and-choose mechanism of Delgado-Segura et al. [5], it explicitly treats semantic assessment as a requester-side activity rather than assuming the existence of an application-specific validation procedure. Finally, our protocol integrates this inspection phase with content-addressed storage, manifest-level cryptographic binding, and zero-knowledge proofs of key possession.

Similarly, ZKCPlus [10], an extension of the Zero-Knowledge Contingent Payment (ZKCP) protocol [11], achieves practical and fair exchange in a trustless manner using the Bitcoin network and ZKPs. ZKCPlus incorporates a new commit-and-prove non-interactive zero-knowledge (CP-NIZK) argument of knowledge under the standard discrete logarithmic assumption, which is prover-efficient for data-parallel computations. In contrast, our work targets data-exchange settings where the relevant notion of quality is semantic, subjective, or too costly to encode as a circuit predicate. Rather than proving full data validity in zero knowledge, we use ZK proofs only for key possession and rely on cut-and-choose sampling for probabilistic semantic assurance.

5 Conclusions and Future Work

This paper presented a decentralized fair data exchange protocol that combines encrypted off-chain data delivery, ECIES-based key wrapping, ZKPs of key possession, and requester-seeded random sampling. The protocol separates cryptographic key validity from semantic usefulness: while the smart contract enforces key-delivery accountability, the requester obtains probabilistic semantic assurance by inspecting randomly selected plaintext chunks before final settlement.

Future work includes an end-to-end implementation, evaluation of proof-generation and on-chain verification costs, formal analysis of the sampling guarantees, and private verifiable delivery for encrypted final key release with automated settlement. In addition, we plan to study the trade-off between sampling ratio, semantic assurance, and information disclosure under different threat models.

References

[1] N. Asokan, V. Shoup, and M. Waidner. Asynchronous protocols for optimistic fair exchange. In *1998 IEEE Symposium on Security and Privacy*, pages 86–99, 1998.

[2] Feng Bao, R.H. Deng, and Wenbo Mao. Efficient and practical fair exchange protocols with off-line TTP. In *1998 IEEE Symposium on Security and Privacy*, pages 77–85, 1998.

[3] Juan Benet. IPFS - Content Addressed, Versioned, P2P File System. *arXiv preprint arXiv:1407.3561*, 2014.

[4] Changhao Chenli, Wenyi Tang, Hyeonbum Lee, and Taeho Jung. Fair^{2}2Trade: Digital Trading Platform Ensuring Exchange and Distribution Fairness. *IEEE Transactions on Dependable and Secure Computing*, 21(5):4827–4842, 2024.

[5] Sergi Delgado-Segura, Cristina Pérez-Solà, Guillermo Navarro-Arribas, and Jordi Herrera-Joancomartí. A fair protocol for data trading based on bitcoin transactions. *Future Generation Computer Systems*, 107:832–840, 2020.

[6] Sergi Delgado-Segura, Cristina Pérez-Solà, Jordi Herrera-Joancomartí, and Guillermo Navarro-Arribas. Bitcoin private key locked transactions. *Information Processing Letters*, 140:37–41, 2018.

[7] Stefan Dziembowski, Lisa Eckey, and Sebastian Faust. FairSwap: How To Fairly Exchange Digital Goods. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 967–984, New York, NY, USA, 2018. Association for Computing Machinery.

[8] Christos Karapapas, Iakovos Pittaras, George C. Polyzos, and Constantinos Patsakis. Hello, won't you tell me your name?: Investigating anonymity abuse in IPFS. In Bart Coppens, Bruno Volckaert, Vincent Naessens, and Bjorn De Sutter, editors, *Availability, Reliability and Security*, pages 187–204, Cham, 2025. Springer Nature Switzerland. (Proceedings ARES, Ghent, Belgium, August 2025).

[9] Yan Li, Lingyan Li, Yanqi Zhao, Nadra Guizani, Yong Yu, and Xiaojiang Du. Toward decentralized fair data trading based on blockchain. *IEEE Network*, 35(1):304–310, 2021.

[10] Yun Li, Cun Ye, Yuguang Hu, Ivring Morpheus, Yu Guo, Chao Zhang, Yupeng Zhang, Zhipeng Sun, Yiwen Lu, and Haodi Wang. ZKCPlus: Optimized Fair-exchange Protocol Supporting Practical and Flexible Data Exchange. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 3002–3021, New York, NY, USA, 2021. Association for Computing Machinery.

[11] Gregory Maxwell. Zero Knoweldge Contingent Payment, 2011. (The author and date are not mentioned in the main page, but are shown in the history.).

[12] Ertem Nusret Tas, István András Seres, Yinuo Zhang, Márk Melczer, Mahimna Kelkar, Joseph Bonneau, and Valeria Nikolaenko. Atomic and Fair Data Exchange via Blockchain. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 3227–3241, New York, NY, USA, 2024. Association for Computing Machinery.

[13] G. Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151, 2014.

[14] Hao Zeng, Helei Cui, Man Li, Bo Zhang, Chengjun Cai, Zhiwen Yu, and Bin Guo. Decentralized and fair trading via blockchain: The journey so far and the road ahead. *IEEE Transactions on Dependable and Secure Computing*, 22(4):4379–4397, 2025.