Secure Decentralized Data Marketplace for AI/ML Datasets

Samsung Prism

May 18, 2025

Abstract

In the era of big data, modern data marketplaces have received much attention as they allow not only large enterprises, but also individuals, to trade their data. This new paradigm makes the data prone to various threats, including piracy, illegal reselling, tampering, illegal redistribution, ownership claiming, forgery, theft, misappropriation, etc. This report examines a decentralized marketplace implementation hat facilitates the secure and transparent exchange of datasets focusing on secure data storage using IPFS (InterPlanetary File System). The system utilizes advanced cryptographic techniques including elliptic curve cryptography (specifically secp256k1), proxy re-encryption, and digital watermarking to ensure data confidentiality, controlled access, and ownership verification. By leveraging blockchain wallet signatures as proxies for private keys, the system maintains security while enabling convenient data access control without exposing sensitive cryptographic material. The implementation enables users to upload, purchase, and transfer datasets with cryptographic ownership verification and tamper resistance. This paper discusses the underlying architecture, challenges faced in decentralized data commerce, and how our proposed solutions create a secure, user-centric platform for dataset monetization and reuse in AI/ML workflows.

1 Introduction

Recent years have witnessed a dramatic increase in the generation of big data due to adoption of new technologies. Most of the enterprises now-a-days consider big data as a most significant resource and harness its power as a drivingforce to their business growth. This evergrowing demands of big data in rapidly changing competitive environment offers a new paradigm which encourages enterprises to adopt data monetization and initiates the establishment of large number of start-ups companies who sell and purchase our personal data on a daily basis. Few, among many others, include Datacamp, Datawallet, Dawex, etc. 1 There are many other situations where data monetization is an integral and indispensable part of a system in the form of data-as-a-service model (Terzo et al., 2013). Some interesting fields spawned and co-existing with the use of big data are machine learning, deep learning, artificial intelligence, data-science, etc., which may demand training dataset in a pay-per-use fashion.

Although the above paradigm shift empowers individuals or organizations to sell their own data, this raises major concerns related to our fundamental right to privacy and data security. In fact, data become prone to various threats, such as piracy, and illegal reselling.

The emergence of blockchain technology and decentralized storage solutions like IPFS has enabled new approaches to data exchange that promise greater security, transparency, and control for all participants. This report analyzes a decentralized marketplace implementation that combines blockchain technology, IPFS storage, and advanced cryptographic techniques to create a secure platform for data exchange. The system addresses several critical challenges in decentralized data commerce:

1. Secure storage of potentially sensitive datasets
2. Controlled access to data based on purchase transactions
3. Ownership verification and protection against unauthorized redistribution
4. Integration with blockchain wallets for authentication and payment

The implementation uses elliptic curve cryptography, specifically the secp256k1 curve (used by Bitcoin and Ethereum), along with proxy re-encryption techniques to enable secure data sharing without exposing private keys. Additionally, it employs digital watermarking to embed ownership information in datasets, allowing for verification of legitimate ownership and detection of unauthorized copies. By examining this implementation in detail, this report provides insights into the technical foundations of secure decentralized data exchange and highlights innovative approaches to common security challenges in this domain.

2 Related Works

2.1 Decentralized Storage Systems

Decentralized storage systems like IPFS (InterPlanetary File System), Filecoin, and Swarm have emerged as alternatives to traditional centralized cloud storage. These systems distribute data across a network of nodes, providing redundancy and resistance to censorship. IPFS, in particular, has gained popularity for its content-addressed storage model, where data is referenced by its cryptographic hash rather than its location.

2.2 Blockchain-Based Marketplaces

Numerous blockchain-based marketplaces have been proposed and implemented for various digital assets, including NFTs (Non-Fungible Tokens), digital art, and data. These marketplaces leverage smart contracts to automate transactions and enforce access control rules. Projects like Ocean Protocol, Filecoin, and Streamr have focused specifically on creating decentralized data marketplaces.

2.3 Proxy Re-encryption

Proxy re-encryption (PRE) was first introduced by Blaze, Bleumer, and Strauss in 1998 and has since been extended and improved by numerous researchers. PRE allows a third party (proxy) to transform a ciphertext encrypted under one key into a ciphertext that can be decrypted by another key, without the proxy learning the plaintext. Notable implementations include NuCypher (now Threshold Network) and ECIES (Elliptic Curve Integrated Encryption Scheme).

2.4 Digital Watermarking

Digital watermarking techniques have been extensively studied for copyright protection of digital media. For structured data like databases and datasets, various approaches have been proposed, including:

- LSB (Least Significant Bit) modification
- Statistical watermarking
- Structural watermarking

Recent work has focused on combining watermarking with blockchain technology to create verifiable proofs of ownership for digital assets.

3 Preliminaries

3.1 Elliptic Curve Cryptography

An elliptic curve is a mathematical object defined by an equation of the form:

$$y^2 = x^3 + ax + b$$

Where a and b are constants that define the specific curve. The curve has some fascinating properties that make it useful for cryptography. The implementation uses the secp256k1 elliptic curve, which is the same curve used by Bitcoin and Ethereum.

$$y^2 = x^3 + 7$$

Key properties of elliptic curves that make them useful for cryptography include:

1. Point Addition: The ability to "add" two points on the curve to get a third point through a geometric operation.
2. Point Doubling: The ability to "double" a point by drawing a tangent line.
3. Scalar Multiplication: The ability to multiply a point by a scalar through repeated addition.
4. Discrete Logarithm Problem: Given points P and Q where Q = nP, it's computationally infeasible to find n.

The secp256k1 curve is preferred for its:

- Computational efficiency
- Strong security with 256-bit keys
- Widespread use in cryptocurrencies (Bitcoin, Ethereum)
- Lack of known vulnerabilities

3.2 Encapsulation

Encapsulation combines public key cryptography with symmetric key cryptography to achieve efficient and secure data exchange. The process involves:

1. Key Encapsulation: Using the recipient's public key to generate a shared secret and a "capsule" containing recovery information.
2. Data Encryption: Using the symmetric key to encrypt the actual data.
3. Transmission: Sending both the capsule and encrypted data to the recipient.
4. Key Recovery: Using the recipient's private key to recover the symmetric key.
5. Data Decryption: Using the recovered symmetric key to decrypt the data.

The encapsulation process goes as follows:

1. Generate two random key pairs (kp1, kp2)
2. Calculate a hash of their public keys. How this works: Input: Array of points on the curve Process: Convert each point to bytes. Convert bytes to hex. Update the hash with each hex string. Get the final hash digest. Convert to a big number. Add 1 to ensure non-zero.
3. Calculate part_S $= sk1 + sk2 \times$ hash
4. Calculate the symmetric key
$$= \text{SHA256}(pk \times (sk1 + sk2))$$
5. Create a capsule with (pk1, pk2, partS)

3.3 Proxy Re-encryption

Proxy re-encryption allows a semi-trusted proxy to transform ciphertexts from one public key to another without accessing the plaintext. The process involves:

1. Re-encryption Key Generation: Creating a special key that allows transformation from owner to buyer.
2. Re-encryption: Transforming the capsule to be decryptable by the buyer.
3. Re-encrypted Decapsulation: Allowing the buyer to recover the symmetric key using their private key.

3.3.1 The Re-encryption Process

1. Generate a random key pair (kp)
2. Calculate a hash of $(tmp\_pk, pk_B, pk_B \times tmp\_sk)$
3. Calculate $rk = sk_A \times \frac{1}{\text{hash}}$
4. Transform the capsule: $(E \times rk, V \times rk, S, tmp\_pk)$

3.3.2 The Decapsulation Process

For original capsules: symmetric_key

$$= \text{SHA256}((E + V) \times sk)$$

For re-encrypted capsules: symmetric_key

$$= \text{SHA256}((E' + V') \times \text{hash} \times sk_B)$$

3.4 Signature Generation (ECDSA)

When a user signs a message, they use the Elliptic Curve Digital Signature Algorithm (ECDSA). The signature consists of two components: (r, s). These are large numbers derived from:

- The message being signed
- The signer's private key
- A random nonce

$$\begin{aligned} s &= k^{-1}(\text{hash}(\text{message}) + d \times r) \mod n \\ r &= (k \times G).x \mod n \\ \text{Signature} &= (r, s) \end{aligned}$$

The signature is typically represented as a hexadecimal string

3.5 IPFS

IPFS (InterPlanetary File System) is a protocol and peer-to-peer network for storing and sharing data in a distributed file system. It uses content-addressing rather than location-addressing, meaning files are identified by their content rather than where they're stored. This provides several advantages:

- Content cannot be changed without changing its address
- Redundancy and availability across multiple nodes
- Resistance to censorship and single points of failure

3.6 Dapp Development

Frontend: React.js framework for building responsive, component-based user interfaces. Integrated with Tailwind CSS for modern design and WalletConnect/MetaMask for wallet interactions.
Backend: Node.js and Express.js for building a scalable RESTful API. It handles user metadata, transaction logs, dataset indexing, and smart contract interaction.
Blockchain: Ethereum is used for executing smart contracts written in Solidity. These contracts manage dataset ownership, access rights, pricing, transactions, and re-encryption metadata.
Database: MongoDB for storing off-chain metadata, user profiles, reviews, dataset filters, and indexing data to facilitate search and filtering operations.

4 Challenges in Decentralized Data Marketplaces

4.1 Private Key Security

In blockchain applications, private keys are extremely sensitive and should never be exposed. Traditional proxy re-encryption requires direct access to private keys, which is problematic in a wallet-based architecture where private keys never leave the secure wallet environment.

4.2 Access Control

Controlling access to data in a decentralized system is challenging because there's no central authority to enforce permissions. Any solution must ensure that:

- Only authorized buyers can access data.
- Access can be granted and revoked without re-encrypting the entire dataset.
- The process is efficient even for large datasets.

4.3 Data Ownership and Redistribution Prevention

Once data is sold to a buyer, there's a risk of unauthorized redistribution. The marketplace needs mechanisms to:

- Verify the legitimate owner of a dataset.
- Detect unauthorized copies.
- Verify and trace the ownership lineage of each dataset to prevent forgery and theft.
- Enabling cryptographically secure and verifiable transfer of dataset ownership.

4.4 Integration with Blockchain Wallets

Blockchain wallets are designed for transaction signing, not for encryption operations. Adapting them for use in a secure data marketplace requires innovative approaches to leverage their existing security features.

5 Proposed Solutions

5.1 Signature-Based Key Proxy

The proposed system addresses the challenge of private key security through an innovative signature-based key proxy mechanism. Instead of directly using private keys for encryption operations, which would compromise wallet security, the system leverages digital signatures as secure proxies for private key operations. This approach maintains the integrity of wallet security while enabling the necessary cryptographic operations for secure data exchange. When cryptographic operations are required, the wallet creates a signature of a specific message using the Elliptic Curve Digital Signature Algorithm (ECDSA). From this signature, the system extracts the first 64 characters to serve as a proxy key, which is then utilized for cryptographic operations without exposing the underlying private key. The ECDSA signature consists of two components (r, s), where r = (k × G).x mod n and s = k^{-}1$(hash(message) + dr)modn$. In these equations, k represents a random nonce, G denotes the base point of the curve, d is the private key, and n represents the order of the curve. This approach effectively bridges the gap between blockchain wallet security requirements and the functional needs of proxy re-encryption systems.

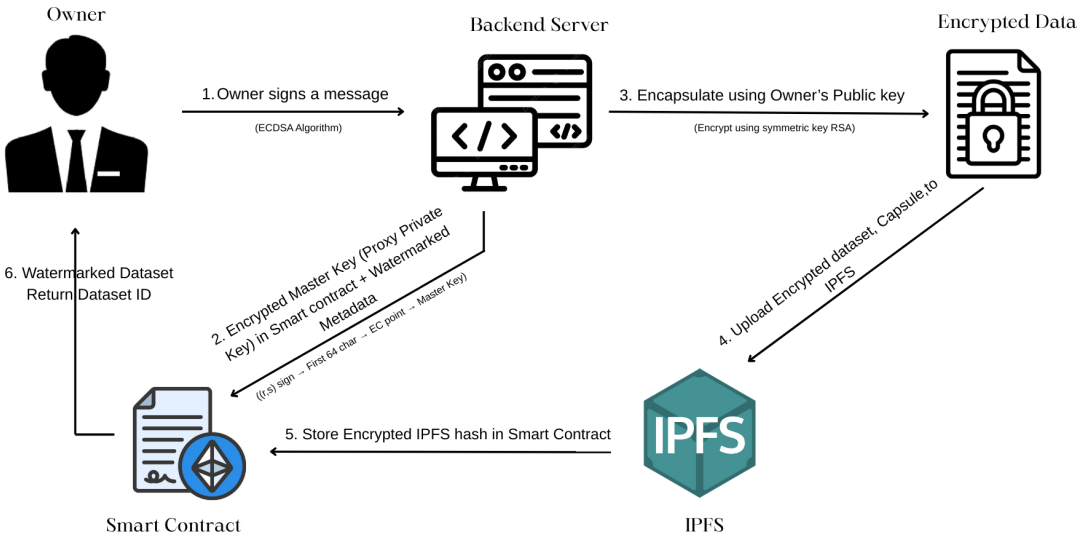

Figure 1: Upload process

5.2 Three-Phase Secure Data Exchange

The secure data exchange process in the proposed system follows a comprehensive three-phase approach that ensures data security throughout the lifecycle of a dataset transaction. During the initial upload phase, data owners begin by signing an authorization message using their wallet. The system extracts a proxy key from this signature and converts it to an elliptic curve point. Through scalar multiplication, the system derives a master key that will facilitate future access control operations. The owner's public key is then used for encapsulation, generating a symmetric key that encrypts the actual data. The system stores the encrypted data, the encapsulation capsule, and the master key, with appropriate distribution between on-chain and off-chain storage based on size and sensitivity considerations.

The purchase process represents the second phase of the data exchange workflow. When a buyer purchases access to a dataset, the system generates a transformation key by combining the previously stored master key with the buyer's public key. This transformation key enables the re-encryption proxy to transform the original capsule into a new form that can be decrypted specifically by the buyer's private key, without requiring re-encryption of the entire dataset or exposing either the owner's or buyer's private keys to the system. The transformation occurs without revealing the underlying data or keys, maintaining the security and integrity of the cryptographic process.

The data access phase completes the secure exchange process. To access the purchased data, the buyer signs an access request message with their wallet. The system extracts a proxy key from this signature and converts it to a decryption key component. By combining the buyer's private key (via the proxy mechanism) with the transformed capsule, the system performs decapsulation to recover the symmetric key originally used to encrypt the data. This symmetric key then enables the decryption of the dataset, providing the buyer with access to the purchased data. This phased approach ensures that at no point are private keys exposed, while still enabling secure transfer of encryption capabilities from owner to buyer.

5.3 Watermarking for Ownership Verification

The system incorporates a sophisticated selective bit-modification watermarking technique to address challenges related to data ownership verification and unauthorized redistribution prevention. This

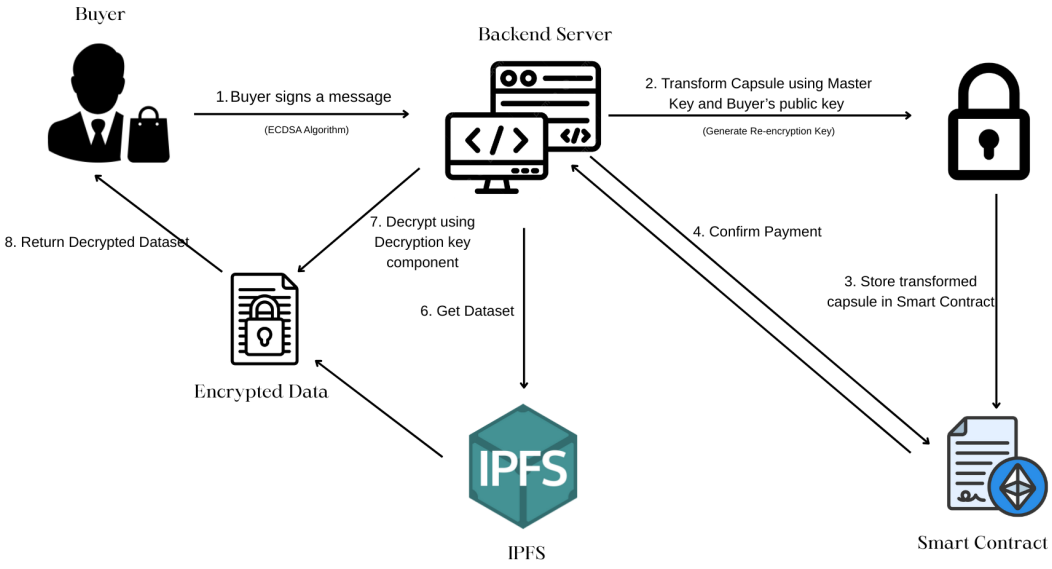

Figure 2: Purchase process

watermarking approach modifies selected bits in the dataset in a deterministic manner based on the owner's identity, creating a unique fingerprint that can be verified without comparison to the original data. The implementation selectively marks approximately 40 percent of data rows (controlled by a GAMMA parameter set to 0.4), focusing modifications on the two least significant bits of selected attributes. This selective approach ensures that the watermark minimally impacts data utility while remaining detectable.

A key strength of the implemented watermarking technique is its deterministic nature, ensuring that the same owner key will produce identical watermarks when applied to the same dataset. This property enables verification of ownership claims without requiring access to an unwatermarked reference copy. The technique is also designed to be tamper-resistant, as modifications to the data will affect the watermark, allowing detection of unauthorized alterations. For example, in a typical modification, a value such as 200 (binary: 11001000) might be subtly altered to 201 (binary: 11001001) by changing only the least significant bit, a change unlikely to significantly impact analytical results but sufficient to encode ownership information across the dataset.

Additionally, the system supports metadata watermarking as an alternative approach that preserves the integrity of the original data entirely while still providing robust ownership verification mechanisms. The integration of blockchain wallet addresses as watermark identifiers creates a direct link between data ownership claims and blockchain identities, facilitating automatic verification and leveraging existing wallet infrastructure.

5.4 Hybrid Storage Architecture

The proposed solution implements a carefully designed hybrid storage architecture that optimizes for both security and efficiency in a decentralized environment. Rather than attempting to store entire datasets on the blockchain—which would be prohibitively expensive and inefficient—the system distributes different components across appropriate storage mechanisms. The encrypted content itself is stored on IPFS (InterPlanetary File System), taking advantage of its content-addressed storage model that inherently provides integrity verification through cryptographic hashing. The system references this off-chain content using its IPFS hash, which is stored in the blockchain's smart contract structure.

The blockchain component of the storage architecture maintains essential metadata about the dataset, including comprehensive file information (name, type, size, and upload date), ownership details that link the data to specific blockchain identities, watermark verification information, and type-specific metadata that enhances discoverability and usability of the dataset. This metadata is structured as JSON data within the smart contract, making it easily accessible and verifiable through blockchain queries. The access control mechanisms are implemented directly through the smart contract, maintaining critical information such as the dataset's price, current availability status, and a mapping of addresses with granted access permissions. This approach leverages the immutability and transparency of blockchain technology for access control while avoiding the overhead of storing large data on-chain. The smart contract thus serves as both an access control mechanism and a registry that links to the off-chain encrypted content, creating a cohesive system that balances security, efficiency, and decentralization requirements.

5.5 Enhanced User Experience and Review System

To improve discoverability, the system allows users to tag datasets with keywords, categories, data types, and application domains. An off-chain search engine (powered by MongoDB with full-text indexing) supports faceted searches and filtering by type, size, price, and reviews. Public datasets can be browsed freely, and any dataset's purchase history, reviews, and current availability are transparently visible. Each review is cryptographically linked to a verified purchase, ensuring authenticity. This promotes a reputation system for both datasets and users.

5.6 Ownership Transfer and Provenance

Ownership transfer is implemented as a state change in the dataset's smart contract entry. The previous owner triggers a transfer function, which records the transaction, updates the ownership field, and logs the timestamp. All historical transactions and previous owners are permanently available on-chain, enabling full provenance tracing.

Upon transferring ownership, the master key—used to encrypt the dataset with a symmetric algorithm—is invalidated to prevent unauthorized access by the new owner using the previous key encapsulation. Rather than re-encrypting and re-uploading the dataset itself to decentralized storage (e.g., IPFS), the architecture relies on re-encryption of the master key and regeneration of cryptographic capsules associated with each buyer. This allows the dataset, which remains securely stored and encrypted under a stable symmetric key, to be re-accessed by authorized purchasers through newly generated capsules that bind the new owner's cryptographic identity with each buyer's public key. This approach ensures continuity of access without redundant storage operations, while preserving strict access control and end-to-end data confidentiality. This is crucial for ensuring trust in high-value datasets used in sensitive AI/ML applications.

6 Security Analysis

6.1 Access Control Security

The system provides strong access control through multiple security layers:

- Point Multiplication: Security relies on the difficulty of the elliptic curve discrete logarithm problem (ECDLP).
- One-way Function: Easy to derive buyer keys from the master key, but hard to recover the master key from buyer keys.
- Private Key Isolation: Private keys never leave wallets, preventing exposure.

6.2 Signature Security

The signature-based approach provides several security benefits:

- One-time Use: Signatures are used only once, preventing replay attacks.

- Message Binding: Signatures include the dataset ID, ensuring they're valid only for specific datasets.

6.3 Data Integrity

- Proxy re-encryption allows access delegation without raw data exposure.
- Watermarking ensures tamper-evident datasets with cryptographic traceability.

7 Smart Contract Functions

```
struct Dataset {
        address owner;
        string metadata;        // IPFS hash for dataset metadata
        string  IpfsHash;    // IPFS hash for encrypted dataset
        string watermarkHash;          // For ownership tracking
        string encryptedMasterKey; // Master key encrypted with owner's
        public key
        uint256 price;
        bool isAvailable;
        uint256 timestamp;
        uint256 accessCount;
        mapping(address => bool) hasPurchased;
        mapping(address => string) transformedCapsules; // Capsules re-encrypted for buy
        mapping(address => string) userWatermarks;
        address[] previousOwners;
        uint256[] transferTimestamps;
    }

function purchaseDataset(uint256 _datasetId) public payable {
        Dataset storage dataset = datasets[_datasetId];

        // Validation checks
        require(dataset.isAvailable, "Dataset is not available");
        require(msg.value >= dataset.price, "Insufficient payment");
        require(!dataset.hasPurchased[msg.sender], "You already purchased this dataset")
        require(msg.sender != dataset.owner, "You can't purchase your own dataset");

        // Mark as purchased
        dataset.hasPurchased[msg.sender] = true;
        ds.accessCount++;

        // Transfer payment to owner
        payable(dataset.owner).transfer(msg.value);

        emit DatasetPurchased(_datasetId, msg.sender);
    }

function storeTransformedCapsule(uint256 _datasetId, address _buyer, string memory _caps
        Dataset storage ds = datasets[_datasetId];
        require(msg.sender == ds.owner || isAuthorizedOracle(msg.sender), "Not authorize
        require(ds.hasPurchased[_buyer], "Buyer hasn't purchased");

        ds.transformedCapsules[_buyer] = _capsule;
        emit CapsuleTransformed(_datasetId, _buyer);
    }
```

```
function updateDataset(uint256 _datasetId, string memory _metadata, uint256 _price) publ
        Dataset storage ds = datasets[_datasetId];
        ds.metadata = _metadata;
        ds.price = _price;
        emit DatasetUpdated(_datasetId, _metadata, _price);
    }

function removeDataset(uint256 _datasetId) public onlyOwner(_datasetId) {
        datasets[_datasetId].isAvailable = false;
        emit DatasetRemoved(_datasetId);
    }

function getDatasetDetails(uint256 _datasetId) public view returns (
        address owner,
        string memory ipfsHash,
        string memory metadata,
        uint256 price,
        bool isAvailable,
        uint256 accessCount,
        bool hasPurchased
    ) {
        Dataset storage ds = datasets[_datasetId];
        return (
            ds.owner,
            ds.ipfsHash,
            ds.metadata,
            ds.price,
            ds.isAvailable,
            ds.accessCount,
            ds.hasPurchased[msg.sender]
        );
    }

function transferOwnership(uint256 _datasetId, address newOwner) public onlyOwner(_datas
        require(newOwner != address(0), "Invalid new owner");
        Dataset storage ds = datasets[_datasetId];
        ds.previousOwners.push(ds.owner);
        ds.transferTimestamps.push(block.timestamp);
        address oldOwner = ds.owner;
        ds.owner = newOwner;
        emit OwnershipTransferred(_datasetId, oldOwner, newOwner, block.timestamp);
    }

function isAuthorizedOracle(address _oracle) internal pure returns (bool) {
        return _oracle == address(0x1234567890123456789012345678901234567890);
    }
}
```

8 Conclusion

The decentralized marketplace implementation demonstrates an innovative approach to secure data exchange using blockchain technology, IPFS, and advanced cryptographic techniques. By leveraging elliptic curve cryptography, proxy re-encryption, and digital watermarking, the system addresses key challenges in decentralized data commerce.

The signature-based key proxy approach allows cryptographic operations without exposing private keys, bridging the gap between blockchain wallet security and proxy re-encryption requirements. The three-phase secure data exchange process ensures security throughout the dataset lifecycle. Combined with watermarking techniques, this forms a comprehensive solution for secure data commerce in a decentralized environment.

Through seamless wallet integration, user-driven metadata, searchable indexing, and transparent transaction logging, it establishes a self-sustaining ecosystem where data can be owned, reviewed, bought, and resold without intermediaries. This marketplace paradigm shift empowers data owners, ensures trust, and paves the way for decentralized AI innovation.

Future work may focus on:

- Optimizing performance for large datasets.
- Enhancing watermarking techniques.
- Implementing more sophisticated access control mechanisms.
- Improving dispute resolution processes.

Overall, decentralized technologies hold the potential to transform data commerce by providing security, transparency, and control without relying on trusted intermediaries.

9 References

References

[1] Blaze, M., Bleumer, G., & Strauss, M. (1998). Divertible protocols and atomic proxy cryptography. In *International Conference on the Theory and Applications of Cryptographic Techniques* (pp. 127-144). Springer.

[2] Benet, J. (2014). IPFS - Content Addressed, Versioned, P2P File System. *arXiv preprint arXiv:1407.3561*.

[3] Brown, D. R. L. (2010). SEC 2: Recommended Elliptic Curve Domain Parameters. Certicom Research.

[4] Johnson, D., Menezes, A., & Vanstone, S. (2001). The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security, 1*(1), 36-63.

[5] Ateniese, G., Fu, K., Green, M., & Hohenberger, S. (2006). Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security (TISSEC), 9*(1), 1-30.

[6] Cox, I. J., Miller, M. L., Bloom, J. A., Fridrich, J., & Kalker, T. (2007). Digital watermarking and steganography. Morgan Kaufmann.

[7] Ethereum. (2016). EIP-191: Signed Data Standard. Ethereum Improvement Proposals. https://eips.ethereum.org/EIPS/eip-191.

[8] Buterin, V. (2014). Ethereum: A next-generation smart contract and decentralized application platform. White paper.

[9] Bernstein, D. J. (2006). Curve25519: new Diffie-Hellman speed records. In *International Workshop on Public Key Cryptography* (pp. 207-228). Springer.

[10] Nu Cypher. (2017). Proxy Re-Encryption for Distributed Systems. White paper.