# ENCODING AND DECODING FUNCTIONAL SCHEMES USING BLOCK CHAIN

**Anusha nallanagula[1],Dr. Deepak sukheeja[2],Dr. Bvkiranmayee[3]**

[1] Student, Department of CSE,Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering &Technology,Bachupally, Hyderabad, Telangana, India
*E-mail:nallanagulaanusha444@gmail.com*

[2] Associate Professor, Department of CSE ,Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering &Technology,Bachupally, Hyderabad, Telangana, India.

[3] Professor, Department of CSE,Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering &Technology,Bachupally, Hyderabad, Telangana, India
*Mail: Kiranmayee…….@ vnrvjiet. in*

## ABSTRACT

**Functional encryption (FE) represents an important advancement beyond traditional public-key encryption since it supports computations on ciphertext without revealing the underlying plaintext. This feature is highly valuable in applications that both require secure storage of information and controlled sharing of information, including cloud storage services. However, the computational intensity of FE schemes presents a major challenge, especially when coherent connectives are supported, leading to execution inefficiency factors. To address this, many proposals suggest that the significant computational tasks should be outsourced to a powerful third-party entity, such as a cloud benefit provider, while keeping the user's tasks minimal. Despite the efficiency gains, the current methods overlook a very important perspective: the requirement for a formalized payment mechanism to reward the mediator for the computational resources consumed. Anticipating cloud administrations to perform seriously computations without an motivating force is unlikely and unsustainable. This work points to the require for a secure and proficient installment system inside useful encryption frameworks, probably by utilizing blockchain-based shrewd contracts to mechanize and secure the remuneration handle. Building up a strong motivation show will improve the common sense of functional encryption in real-world applications, guaranteeing both computational proficiency and financial practicality.**

***Keywords-Functional Encryption, Public-Key Encryption, Data Sharing, Cloud Storage Services, Computational Efficiency, Third-Party Arbitrator, Incentive Model, Payment Mechanism, Blockchain, Smart Contracts, Secure Data Processing.***

## I. INTRODUCTION

Information, in today's developed time, has become one of the most profitable resources. Infinite amounts of client information are managed with the help of cloud capacity administrations in today's developed world. Thus, entrusting delicate data to third-party suppliers raises concerns about security, security, and the misuse of information. These have led to the development of encryption procedures to protect client information, thus ensuring that only authorized parties can access it. FE goes a step further than this by allowing computations on encrypted information without revealing the underlying plaintext, thereby enhancing privacy-preserving information sharing.

Despite its focal points, FE frameworks confront critical challenges in terms of computational complexity. Scrambling and decoding information require resource-intensive operations that may be illogical for gadgets with restricted preparing control. To overcome this, numerous frameworks outsource overwhelming computations to effective third-party substances, such as cloud benefit suppliers. Whereas this arrangement moves forward effectiveness, it presents modern dangers, such as potential information misuse by the referee and expanded benefit costs.

This may be sought after by a decentralized strategy that will enroll normal clients or irregular members in the decoding handle. In this demo, clients will contribute their computational resources to help with the decoding procedure while getting rewards in exchange. Blockchain innovation is a normal fit for overseeing these motivating forces due to its decentralized, straightforward, and secure nature.

The disseminated record of blockchain ensures that all transactions are recorded permanently, awaiting alteration or control. Savvy contracts can automate the payment process, ensuring that supporters are fairly compensated once their tasks are confirmed. This approach eliminates the reliance on a single referee and advances a collaborative, community-driven system for secure information processing.

By combining blockchain with utilitarian cryptography frameworks, the possibility of balance between computational power and financial viability is achieved. Clients can thus share and treat information safely with no compromise towards security, whilst supporters are enticed to become part of the decoding handle. This innovative pairing opens up secure and efficient means of data exchange across various organizations, from business to healthcare and further.

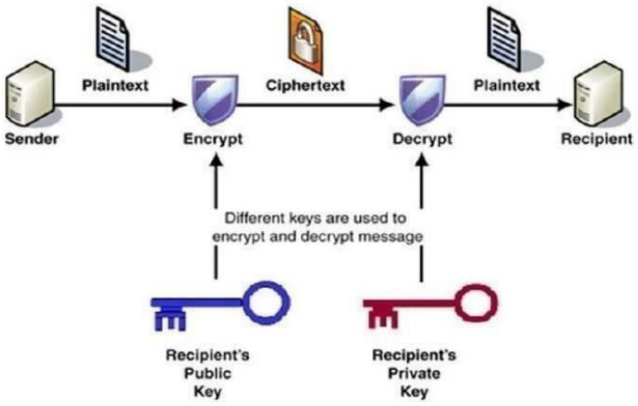


**Fig 1: System Architecture**

## II. RELATED WORK

The sending and interaction with a wise contract on the Ethereum blockchain through Python involves steps in the sense of sending a sharp contract via Truffle. Subsequently, through installing the nearby environment for a blockchain and carrying out the command in truffle to move over the contract to it, then that contract would send and produce the contract's address which is really indispensable for progress and interaction to proceed with this advanced contract. After the contract has been sent, Python is used to interact with it using the web3.py library. Through connection to an Ethereum node, either locally or through Infura, the Python script is able to access the contract sent by using its address and ABI, the Application Parallel Interface. Its abilities can then be called: it can retrieve the client data that is stored, for example, or send transactions in order to store unused data. For illustration, capacities like getUsers can be called utilizing .call(), whereas activities that alter the blockchain, such as storeUser, require marking and sending a exchange utilizing the private key. Furthermore, the shrewd contract can encourage a remunerate framework for members contributing to the unscrambling handle by utilizing blockchain's straightforwardness and permanence for overseeing installments. By allowing members to contribute computing assets for decoding, the smart contract can store details of each instalment exchange and reward supporters securely. The installation part is maintained through Robustness, whereby stores are held into the contract and members get paid according to their contributions. The smart contract

controls the dispersion of installments and ensures the prepare is decentralized and transparent.

| No | Title | Authors | Year | Explanation |
|---|---|---|---|---|
| 1 | On Balanced Codes | S. Al-Bassam and B. Bose | 1990 | Discusses balanced codes for reducing errors by ensuring equal numbers of ones and zeros in codewords. |
| 2 | Design of Efficient Balanced Codes | S. Al-Bassam and B. Bose | 1994 | Introduces efficient methods for designing balanced codes to reduce redundancy in communication systems. |
| 3 | On Unordered Codes | B. Bose | 1991 | Focuses on unordered codes where the order of bits doesn't matter, useful for parallel computing and data compression. |
| 4 | Matched Spectral-Null Codes for Partial-Response Channels | R. Karabed and P.H. Siegel | 1991 | Studies spectral-null codes for reducing distortion in digital communication channels. |
| 5 | Efficient Balanced Codes | D.E. Knuth | 1986 | Discusses efficient balanced codes to ensure data integrity and reduce errors in systems. |
| 6 | Data Integrity in Digital Optical Disks | E.L. Leiss | 1984 | Focuses on maintaining data integrity in optical disk systems like CDs and DVDs. |
| 7 | Error Correcting Codes and Self-Checking Circuits in Fault-Tolerant Computers | D.K. Pradhan and J.J. Stiffler | 1980 | Explains error-correcting codes and self-checking circuits used in fault-tolerant computers to ensure reliability. |
| 8 | High-Order Spectral-Null Codes-Constructions and Bounds | R.M. Roth, P.H. Siegel, A. Vardy | 1994 | Explores high-order spectral-null codes for minimizing distortion in communication systems. |

## III. IMPLEMENTATION

The sending and interaction with a wise contract on the Ethereum blockchain through Python involves steps in the sense of sending a sharp contract via Truffle. Subsequently, through installing the nearby environment for a blockchain and carrying out the command in truffle to move over the contract to it, then that contract would send and produce the contract's address which is really indispensable for progress and interaction to proceed with this advanced contract. After the contract has been sent, Python is used to interact with it using the web3.py library. Through connection to an Ethereum node, either locally or through Infura, the Python script is able to access the contract sent by using its address and ABI, the Application Parallel Interface. Its abilities can then be called: it can retrieve the client data that is stored, for example, or send transactions in order to store unused data. For illustration, capacities like getUsers can be called utilizing .call(), whereas activities that alter the blockchain, such as storeUser, require marking and sending a exchange utilizing the private key. Furthermore, the shrewd contract can encourage a remunerate framework for members contributing to the unscrambling handle by utilizing blockchain's straightforwardness and permanence for overseeing installments. By allowing members to contribute computing assets for decoding, the smart contract can store details of each instalment exchange and reward supporters securely. The installation part is maintained through Robustness, whereby stores are held into the contract and members get paid according to their contributions. The smart contract controls the dispersion of installments and ensures the prepare is decentralized and transparent.

## IV. ALGORITHM

### 1. Functional Encryption (Fe) Algorithm

functional encryption (fe) allows us to encrypt data (make it unreadable) and then perform calculations on the encrypted data without revealing the original information. this is useful when we want to securely store data but still be able to work with it.

**Key Generation:** We generate a public key (pk) to encrypt the data and a private key (sk) to decrypt it.

**Encryption:** Data is encrypted using the public key. for example, if x is the original data, the ciphertext is created

$$C = E_{pk}(x)$$

Perform computation on ciphertext: a function f is applied to the encrypted data. the data remains encrypted, but we can still compute certain values from it.

f(c) (function applied to ciphertext)

**Decryption:** After computing, we use the private key (sk) to decrypt the result and get the answer.

### 2. Blockchain Smart Contract for Payment Mechanism

The blockchain is used to automatically manage payments for cloud service providers that perform computations on the encrypted data. A smart contract ensures that the payment happens securely and automatically.

### 3. Cloud Service Provider Interaction

The cloud provider (a third-party) does the heavy lifting of computations on the encrypted data. They

perform the calculations and then send the results back securely to the user.

**Upload Encrypted Data:** The user uploads encrypted data to the cloud service provider.

**Perform Computation:** The cloud provider uses Functional Encryption to perform computations on the encrypted data.

**Send Back Results:** After the computation is done, the cloud provider sends the results back to the user in encrypted form.

**Payment via Smart Contract:** Once the task is completed, the smart contract automatically processes the payment for the cloud provider.

## RESULTS

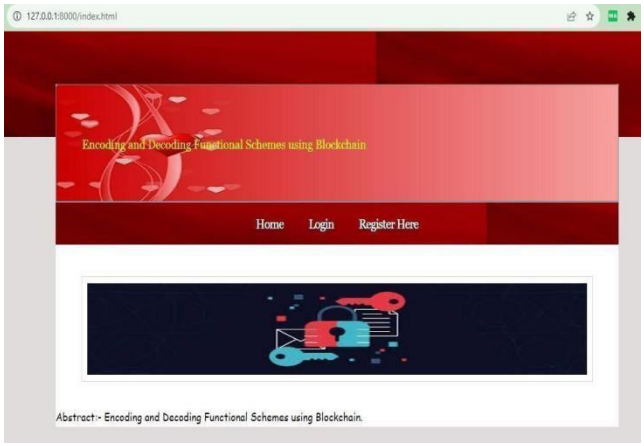


**Fig 1:Home Page**

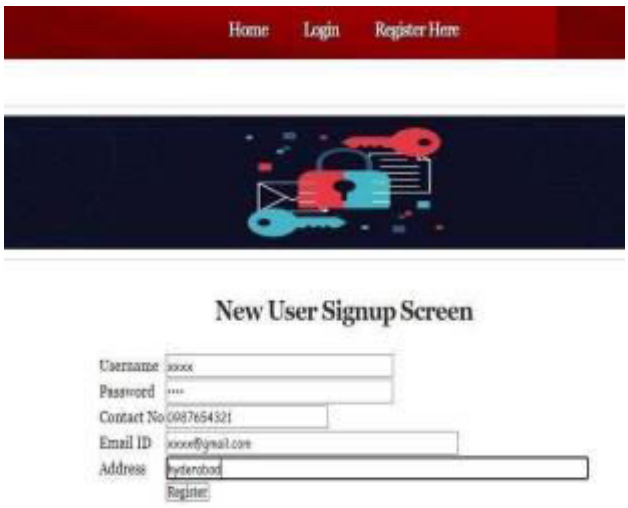


**Fig 2:User Login Screen**

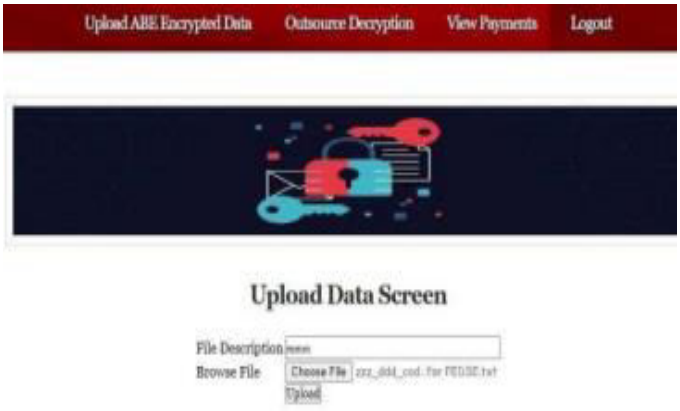


**Fig 3:Uploading document to encrypt**

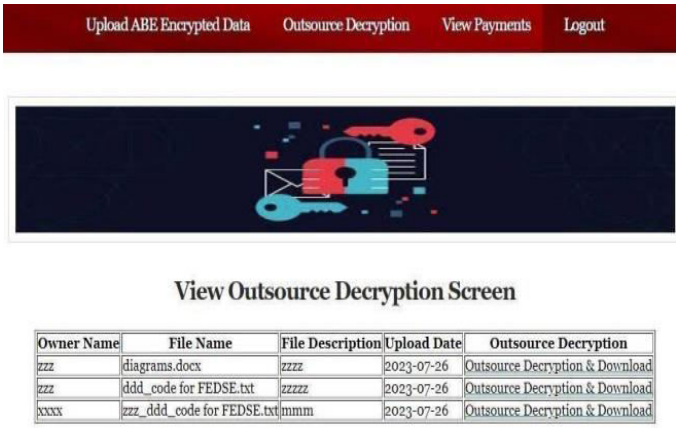


**Fig 4:Encrypted documents list**

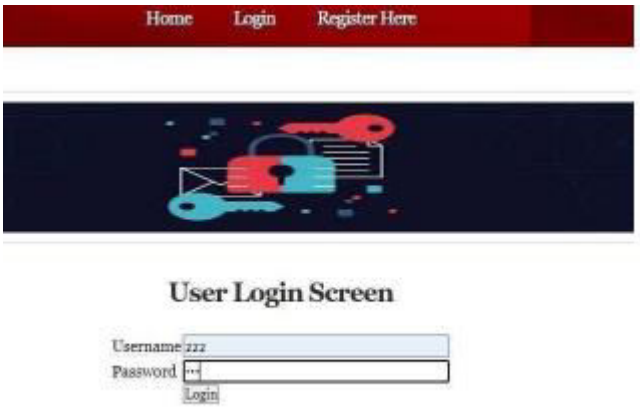


**Fig 5: User Login Screen**

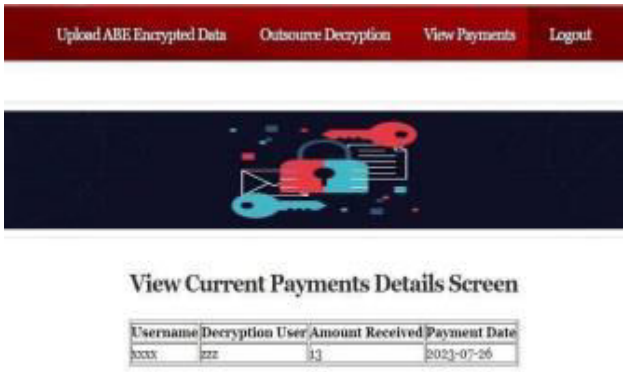


**Fig 6:Payment Screen**

## CONCLUSION

In conclusion, the integration of blockchain innovation with functional encryption speaks to a transformative approach to secure information sharing. This is because it empowers decentralized, straightforward, and incentive-driven decoding forms that answer key challenges in present day encryption frameworks, for example, computational wastefulness and dependence on trusted third parties. Blockchain-based savvy contracts guarantee that supporters are reasonably remunerated for their interest, which cultivates a collaborative environment that upgrades both security and information security.

The implications of this system are vast and cover businesses like healthcare, back, and government, where information sharing is elementary and must be done securely. For example, in healthcare, it can ensure touchy quiet records while allowing authorized computations for restorative inquire about. In back, it guarantees secure exchanges without breaking down client mystery. Government offices can use this show to improve clearness and responsibility in information sharing.

As information security concerns continue to evolve in the computerized age, tried and true encryption with paid outsourcing of decoding offers a down to earth and adaptable arrangement. This system not as it were mitigates the dangers of information abuse but too gives a economical financial demonstrate for secure information preparing. This innovative approach has the potential to revolutionize the way delicate data is shared and prepared in the future by combining the qualities of useful encryption and blockchain.

## REFERENCES

**1.**S. Al-Bassam and B. Bose, "On Balanced Codes", IEEE Trans. Information Theory, vol. 36, pp. 406-408, Mar. 1990.

**2.**S. Al-Bassam and B. Bose, "Design of Efficient Balanced Codes", IEEE Trans. Computers, vol. 43, no. 3, pp. 362-365, Mar. 1994.

**3.**B. Bose, "On Unordered Codes", IEEE Trans. Computers, vol. 40, no. 2, pp. 125-131, Feb. 1991.

**4.**R. Karabed and P.H. Siegel, "Matched Spectral-Null Codes for Partial-Response Channels", IEEE Trans. Information Theory, vol. 37, pp. 818-855, May 1991.

**5.**D.E. Knuth, "Efficient Balanced Codes", IEEE Trans. Information Theory, vol. 32, pp. 51-53, Jan. 1986.

**6.**E.L. Leiss, "Data Integrity in Digital Optical Disks", IEEE Trans. Computers, vol. 33, pp. 818-827, Sept. 1984.

**7.**D.K. Pradhan and J.J. Stiffler, "Error Correcting Codes and Self-Checking Circuits in Fault-Tolerant Computers", IEEE Computer Magazine, vol. 13, pp. 27-37, Mar. 1980.

**8.**R.M. Roth, P.H. Siegel and A. Vardy, "High-Order Spectral-Null Codes-Constructions and Bounds", IEEE Trans. Information Theory, vol. 40, pp. 1826-1840, Nov. 1994.

**9.**S.J. Piestrak, "Design of Self-Testing Checkers for Unidirectional Error Detecting Codes", Scientific Papers of the Inst. of Technical Cybernetics of the Technical Univ. of Wroclaw, no. 24, 1995.