Hybrid Cryptographic System for Secure, Scalable File Encryption and Decryption Using RSA and AES

[1]Mohit Kumar Malviya, [2]Prof. Manish Kumar Singhal
[1]M.tech Scholar, [2]Associate Professor & H.O.D
[1,2]Department of Information Technology (IT)
[1,2]NRI Institute Of Information Science And Technology, Bhopal (MP), India

***Abstract*— Secure and scalable data encryption and decryption have become crucial for protecting sensitive information. Traditional cryptographic systems often face trade-offs between security, efficiency, and scalability. This paper introduces a hybrid cryptographic system that leverages the strengths of both RSA and AES algorithms to address these challenges. RSA, a public-key cryptographic algorithm, offers robust security for key exchange, while AES, a symmetric-key algorithm, provides high-speed data encryption and decryption..The proposed system uses RSA for secure transmission of the AES session key, which is unique for each encryption process. Once securely exchanged, AES is employed for the actual encryption and decryption of files, significantly enhancing performance compared to purely asymmetric encryption. This hybrid approach enables secure file storage and transmission, adaptable to various data volumes and processing speeds. Experimental results demonstrate that the hybrid system achieves efficient encryption and decryption, reducing computational overhead while maintaining high levels of security. This solution is particularly well-suited for cloud storage, secure file-sharing platforms, and environments with high data volume and user demand, addressing the modern needs of scalable, secure file handling.**

***Keywords*—AES (Symmetric Encryption), Chunk-Based Encryption, Distributed Key Management, File Encryption and Decryption, Hybrid Cryptography, RSA (Asymmetric Encryption), Secure Key Exchange**.

I. INTRODUCTION

The Hybrid Cryptographic System combines the strengths of two powerful encryption methods, RSA and AES, to provide a secure, scalable solution for file encryption and decryption. RSA, an asymmetric encryption technique, leverages two separate keys—a public key for encryption and a private key for decryption—to securely distribute AES keys to intended recipients. AES, on the other hand, is a symmetric encryption method known for its speed and efficiency, particularly with large data volumes. By first using RSA to securely encrypt the AES key and then using AES for fast data encryption, this hybrid approach ensures both high-level security and practical performance. The system is designed to protect sensitive information against unauthorized access while also enabling scalability across diverse file sizes and encryption needs. This blend of RSA and AES addresses the limitations of each technique on its own, achieving an ideal balance between security and speed for modern data protection requirements.

In a Hybrid Cryptographic System, RSA handles the secure exchange of keys, mitigating the risk of key interception during transmission. Once the AES key has been securely shared via RSA, AES efficiently encrypts and decrypts large volumes of data, making it highly suitable for scalable applications, such as cloud storage, secure file sharing, and large-scale data processing. The combination also enhances security by separating the key management process from data encryption, reducing the chance of exposing critical information.

This layered security model reinforces the system's robustness, as any compromise in one layer requires breaching the other to access the underlying data. Consequently, hybrid cryptographic systems leveraging RSA and AES meet both security and scalability demands, making them ideal for environments where data privacy and speed are paramount.

The versatility of hybrid cryptography extends further by enabling flexible integration with existing systems and protocols. With RSA managing secure key exchanges and AES offering fast, high-throughput encryption, the hybrid approach ensures that secure data transmission is possible even over potentially insecure networks, such as the internet. In practice, the process typically begins with the sender encrypting the AES key using the recipient's RSA public key, ensuring only the recipient with the corresponding private RSA key can access it. The data itself is encrypted with AES and transmitted alongside the encrypted AES key.

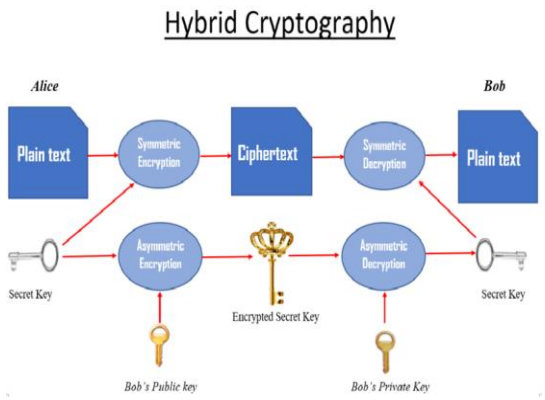

Fig 1 Hybrid Cryptography Model

The system's design also allows for updating or replacing either RSA or AES components independently, making it future-proof and resilient against evolving cryptographic standards and threats. This flexibility has led hybrid cryptographic systems to become widely adopted in industries such as finance, healthcare, and government, where data protection is essential.

II. LITERATURE SURVEY

Chidi Ukamaka Betrand etal [1] Cloud is used in various fields for storage of big data with the major challenge regarding this storage being security. Existing conventional encryption systems can be vulnerable to brute-force attacks. The goal of this project was to develop a hybrid encryption scheme that will only allow authorized users to access and download files stored online, thus enhancing file storage security in the cloud. Rapid Application Development (RAD) methodology was used to create the proposed system, allowing for modifications to be made to the system as it was being developed. The hybrid encryption scheme employs both symmetric and asymmetric encryption. The AES (Advanced Encryption Standard) algorithm and RSA (Rivest–Shamir–Adleman) algorithm were combined to develop the proposed hybrid encryption system. PHP, JavaScript and Laravel were the programming languages and web framework used to implement the system. The proposed system was tested and evaluated by users. The experimental results show that the proposed hybrid encryption scheme was fast and provided a high level of security but had some drawbacks which include increase in file size after it was encrypted and inability to sort files in the web app. Overall, the proposed system enhances confidentiality and data protection in cloud environments, guarding against potential breaches and unauthorized access.

Renuka Shone Durge et al [2] The integration of deep learning with encryption algorithms signifies a groundbreaking advancement in digital security, offering a promising pathway to fortify cyber security measures against the complexity of contemporary cyber threats. This research elucidates the potent synergy between the predictive capabilities of deep learning and the robust security offered by advanced encryption techniques. Through our methodological approach, we demonstrate a dual-layer encryption system that not only enhances data security but also maintains efficiency in data transfer and computational resource allocation. This model presents a significant leap forward in securing sensitive textual data across vulnerable digital channels. Our findings indicate a substantial enhancement in cyber security, suggesting a pivotal shift towards more dynamic and resilient security architectures. This research contributes to both the academic sphere and practical field, laying the groundwork for future innovations in cyber security strategies. It underscores the critical role of emerging technologies in crafting a secure digital ecosystem, capable of withstanding the evolving landscape of cyber threats.

Rajesh Gundla et al [3] The "Hybrid Approach to Cloud Storage Security Using ECC-AES Encryption and Key Management Techniques" algorithm holds the potential to significantly enhance the security, efficiency, and reliability of cloud storage systems. It offers a comprehensive solution that balances cryptographic strength with practical implementation, making it a promising avenue for organizations seeking to secure their cloud-stored data. The proposed hybrid approach combines the strengths of Elliptic Curve Cryptography (ECC) and the Advanced Encryption Standard (AES) to enhance the security of cloud storage systems. By encrypting the AES key using ECC and implementing key splitting and rotation techniques, the algorithm offers a robust solution to protect sensitive data stored in the cloud. Additionally, access control mechanisms and two-factor authentication ensure that only authorized users can access the encrypted data.

Samson Michael Khamis Wani et al [4] The main target of this system is to safely store and recoup data on the cloud that's only controlled by the proprietor of the data. Cloud storage problems of data security are answered using cryptography and steganography ways. Data security is attained using RC6, 3DES, and AES algorithms. Crucial information is efficiently stored using the LSB fashion (Steganography). less time is consumed for the encryption and decryption process using the multithreading fashion. With the help of the proposed security medium, we've fulfilled better data integrity, high security, low detention, authentication, and confidentiality.

III. PROPOSED METHOD

A. System Architecture Overview:

The secure file storage system is designed to offer a robust and efficient mechanism for users to upload, encrypt, store, and later decrypt and restore files. It employs the Flask web framework to create a seamless front-end interface, allowing users to interact with the system intuitively. A key feature of this architecture is its hybrid encryption mechanism, which combines the strengths of both asymmetric and symmetric encryption to safeguard data throughout its lifecycle.

B. File Upload and Handling:

The first step in the secure file storage process is the user uploading a file through the web interface. Users interact with a straightforward and user-friendly interface, enabling them to select files for upload effortlessly. The system performs file validation to check for permissible file types, such as .pem and .txt. This validation ensures that only safe and suitable files are processed, mitigating potential security risks. Once a file is validated, it is temporarily stored in a designated upload directory on the server, which is essential for managing files prior to encryption.

C. File Splitting and Chunk-Based Encryption:

Once the file is uploaded, it undergoes a process of splitting into smaller segments or chunks. This chunking enhances security and improves manageability during encryption and decryption. Large files are divided into smaller, fixed-size chunks, allowing for efficient processing of data. Each chunk can be handled independently, facilitating parallel processing and minimizing memory usage. Additionally, the system creates a metadata file that contains vital information about the original file, including its name and the number of chunks produced. This metadata is crucial for accurately reassembling the file during the decryption phase.

D.Hybrid Cryptography:

The system’s security framework relies on a hybrid cryptographic approach, which utilizes both asymmetric and symmetric encryption techniques. RSA (Rivest-Shamir-Adleman) is implemented for the secure transmission of symmetric encryption keys. RSA employs a public-private key pair, where the public key encrypts the AES key, ensuring that only the holder of the corresponding private key can access it. In parallel, the Advanced Encryption Standard (AES) algorithm is used to encrypt each chunk of the file. AES is preferred due to its high speed and robust security features, making it well-suited for handling large datasets. Each file benefits from a unique AES key, further bolstering security.

E. Encryption Process:

The encryption process is meticulously designed to uphold the highest standards of data security. A random AES key is generated for each file upload, which is essential for the encryption and decryption of the file chunks. Each chunk is then encrypted using this AES key, ensuring that unauthorized access to the chunks does not compromise data integrity, as the chunks cannot be decrypted without the correct AES key. The AES key itself is subsequently encrypted with the RSA public key, ensuring that it can only be decrypted by an individual with the associated RSA private key. This layered encryption approach enhances overall security significantly.

F. Decryption Process:

The decryption process is essentially the inverse of the encryption steps, allowing for the restoration of the original file. To initiate decryption, the system retrieves the RSA-encrypted AES key, which requires the user to provide the private key linked to the public key used during encryption. Once the AES key is decrypted, the system decrypts each encrypted chunk of the file, ensuring that the order of the chunks is preserved as indicated in the metadata. After all chunks have been decrypted, the system reconstructs the original file by combining

the chunks in the correct order, restoring it to its pre-encrypted state.

G. Security Protocols:

The secure file storage system incorporates multiple security measures to protect the integrity and confidentiality of the files. Confidentiality is ensured through the use of AES for chunk encryption, which keeps data unreadable even if intercepted. The additional RSA encryption of the AES key provides an extra layer of security during key transmission. Each chunk is encrypted independently, making any tampering detectable during decryption, as tampered chunks will fail to decrypt properly. The system enforces strict key management practices, keeping private keys secure and avoiding their transmission over the network to minimize interception risks.

H. Error Handling and Logging:

To maintain operational efficiency and facilitate troubleshooting, robust error handling mechanisms are integrated into the system. The system is designed to detect and respond to potential errors, such as unsupported file types or incorrect decryption keys, providing users with informative feedback through the web interface to assist in resolving issues. Additionally, detailed logging mechanisms track all actions within the system, including uploads, downloads, encryption, and decryption attempts. These logs serve critical auditing purposes and are instrumental in identifying any security anomalies or breaches.

I. Testing and Performance Evaluation:

Thorough testing is essential to validate the functionality and performance of the secure file storage system. Performance metrics assess the system's efficiency based on encryption and decryption speed, as well as memory utilization. The implementation of chunk-based processing ensures that performance remains optimal, even when handling large files. Security testing is conducted rigorously to uncover potential vulnerabilities, including unauthorized access attempts to encrypted files and verification of robust encryption measures. The system is designed with scalability in mind, enabling it to efficiently manage increasing data volumes without compromising performance.

J.User Documentation and Training:

To empower users to effectively utilize the system, comprehensive documentation is provided. Detailed user manuals guide users on how to upload files, access encrypted files, and perform decryption processes, ensuring easy navigation within the system. Additionally, training sessions may be organized to familiarize users with the system's features and best practices for maintaining security. This educational component is vital for fostering user confidence and promoting secure file handling practices.

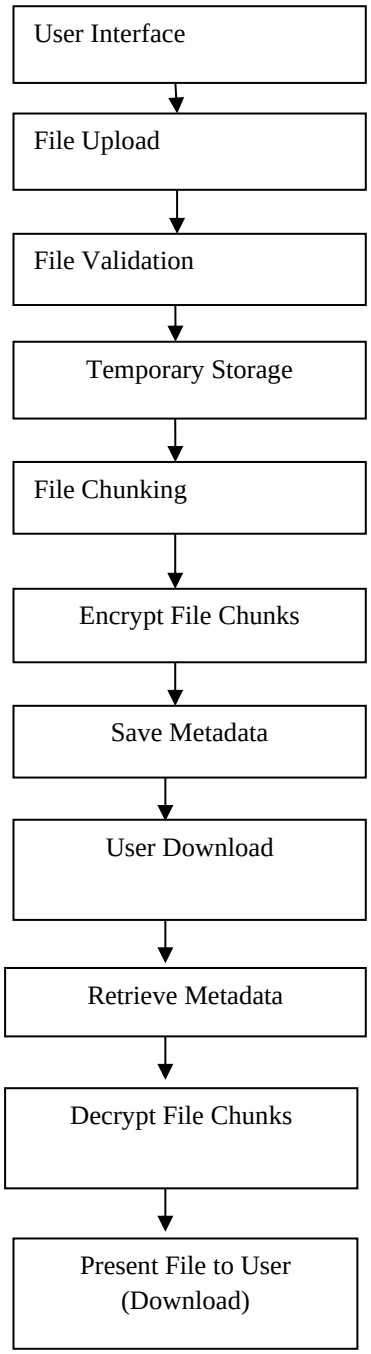

Fig 2 Hybrid Cryptography Model Flow

Algorithm

Start: Initiate program execution.

User Upload Process: Display the file upload interface.
File Validation: Check if the file type is allowed.
Temporary Storage: Save the valid file temporarily on the server.
File Chunking: Split the uploaded file into smaller chunks.
Key Generation: Generate a random AES key for encryption.
Encrypt File Chunks: Encrypt each chunk using the AES key.
Secure Key Handling: Encrypt the AES key with the RSA public key.
File Download Process: Wait for the user to request the file download.
Retrieve the RSA-encrypted AES key: Access the encrypted AES key for decryption.
Request Private Key: Ask the user to provide their RSA private key.
Decrypt the AES key: Use the private key to retrieve the original AES key.
Decrypt Chunks: Decrypt each encrypted chunk with the AES key.
Reassemble the File: Combine decrypted chunks in the correct order.
Display Download Option: Present the user with the option to download the restored file.
Error Handling: Log errors and display messages for issues encountered.
End: Conclude program execution.

IV. SIMULATION AND RESULT

Fig 3 Web-Based File Upload Interface for Secure Submission

A. System Initialization and User Interface

Upon execution, the system initializes the Flask server, exposing a web-based front-end for user interaction. The File Upload Interface is designed to handle multipart form data, enabling users to securely submit files for encryption. The front-end uses Flask's templating engine to render the interface dynamically and handles various file formats (e.g., .txt, .pem).

Fig 4 Validation and Temporary Storage of User Files

B. File Validation and Pre-Processing

Once the user selects a file, it is passed through a validation pipeline where MIME types are verified. This step ensures only permissible file types are accepted, preventing potential exploitation of the file handling process. If valid, the file is temporarily stored on the server using Flask's secure file-handling utilities

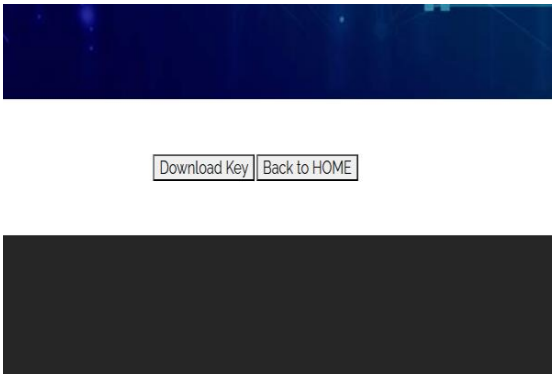

Fig 5 Asymmetric RSA Key Exchange Mechanism for Securing AES Keys

C. File Chunking and AES Encryption

The system splits the validated file into fixed-size chunks using a predefined chunk size parameter (e.g., 1MB). Each chunk is encrypted using the AES symmetric encryption algorithm. For each file upload, a unique AES key is generated and associated with the file metadata. This chunking process allows for parallel encryption, optimizing performance and reducing memory overhead.

D. RSA Key Pair Generation and Key Encryption

The AES encryption key generated during the file chunking stage is encrypted using RSA with a 2048-bit public-private key pair. The RSA-encrypted AES key is stored securely within the system, ensuring that only the corresponding private key can be used to decrypt the AES key, which is crucial for later decryption of the file chunks.

Fig 6 Download of Encrypted File Chunks and RSA-Secured AES Key

E. Encrypted File Download and Metadata Storage

After encryption, the system creates a metadata file containing critical information about the original file and chunk structure. The user is then presented with a download link for the encrypted file, which includes both the encrypted file chunks and the RSA-encrypted AES key. The metadata file ensures accurate reconstruction during decryption.

F. Decrypt method

Fig 7 Web-Based File Upload Interface for decrypt

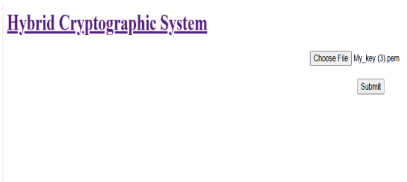

Fig 8 Decrypting AES Key Using RSA Private Key and Decrypting File Chunks

G. Decryption Process Initialization

To initiate decryption, the user uploads the encrypted file and provides the RSA private key. The system retrieves the RSA-encrypted AES key and decrypts it using the provided private key. The AES key is then used to decrypt each encrypted file chunk in sequence, with the original chunk order maintained based on metadata.

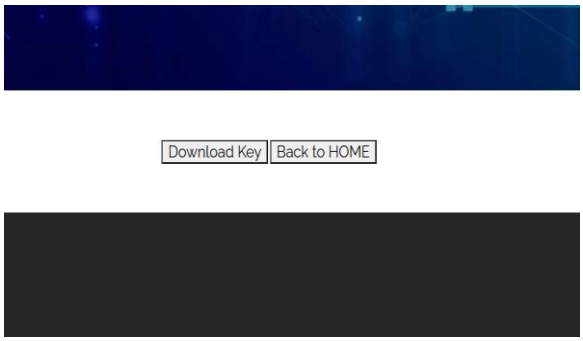

Fig 9 Click on download button to decrypted file download

Fig 10 Reassembling Decrypted Chunks and Providing Original File for Download

H. File Reconstruction and Final Decrypted Output

After successfully decrypting the file chunks, the system reassembles the chunks into their original sequence as specified by the metadata. This process restores the file to its original state, which is then presented for download through the web interface. All decryption steps are logged for auditing purposes, ensuring traceability and error handling in case of failure.

V. CONCLUSION

The secure file storage system developed using a hybrid cryptographic approach successfully provides a robust mechanism for secure file encryption, storage, and retrieval. By combining the efficiency of symmetric encryption (AES) for bulk data handling with the security of asymmetric encryption (RSA) for key exchange, the system ensures both high-performance processing and

strong data protection. The architecture, which splits files into chunks for independent encryption, enhances manageability and performance, particularly for large files. Additionally, chunk-based encryption with associated metadata enables precise reassembly during the decryption process, further ensuring the integrity of the restored file.

The system's modular design and use of Flask as the front-end framework ensure scalability and usability, offering an intuitive interface for users to encrypt and decrypt files with ease. With efficient memory management and chunk-based processing, the system is also optimized for scalability, handling large datasets while maintaining performance

REFERENCES

[1] Chidi Ukamaka Betrand, Chinwe Gilean Onukwugha, Mercy Eberechi Benson-Emenike, Christopher ifeanyi Ofoegbu, Nneka Martina Awaji "File Storage Security in Cloud Computing Using Hybrid Encryption" ISSN: 2376-7731,2024; 12(1): 1-9.

[2] Renuka Shone Durge, Vaishali M. Deshmukh "Advancing cryptographic security: a novel hybrid AES-RSA model with byte-level tokenization"Vol. 14, No. 4, August 2024, pp. 4306~4314,ISSN: 2088-8708.

[3] Rajesh Gundla, Dr. Sunil Gupta,Dr.R Basheer Mohamed "Hybrid Approach To Cloud Storage Security Using ECC- AES Encryption And Key Management Techniques" ISSN: 2832-2754 (Online), Volume-2 Issue-6, Nov-Dec 2022,pp.15-20.

[4] Samson Michael Khamis Wani, Abhay Kumar "Secure File Storage on Cloud Using a Hybrid Cryptography Algorithm"Volume 5, Issue 5, May 2022.

[5] Narendra Shyam Joshi, Kuldeep P. Sambrekar , Abhijit J. Patankar , Archana Jadhav and Prajakta Khadkikar. "Optimizing Encrypted Cloud Data Security and Searchability through Multi-Keyword Ranking Search Methods." ISSN (2210-142X) (2024) Int. J. Com. Dig. Sys. , No. (Mon-20..).

[6] Narendra Shyam Joshi, Kuldeep P. Sambrekar , Abhijit J. Patankar , Archana Jadhav and Prajakta Khadkikar. "Optimizing Encrypted Cloud Data Security and Searchability through Multi-Keyword Ranking Search Methods." International Journal of Computing and Digital Systems, ISSN (2210-142X) (2024).

[7] Le Li 1 , Dong Zheng 1,2, Haoyu Zhang 1 , And Baodong Qin. "Data Secure De-Duplication and Recovery Based on Public Key Encryption With Keyword Search." VOLUME 11, 24 March 2023.

[8] M. Suganya1 and T. Sasipraba. "Comparison and Analysis of Transformer-less Topologies for Grid-Connected PV Systems." Stochastic Gradient Descent long short-term memory based secure encryption algorithm for cloud data storage and retrieval in cloud computing environment, 09 May 2023.

[9] Bijeta Seth, Surjeet Dalal, Vivek Jaglan, Dac-Nhuong Le, Senthilkumar Mohan, Gautam Srivastava. "Integrating encryption techniques for secure data storage in the cloud." Citations: 27,Volume33, Issue404 September 2022.

[10] Udochukwu Iheanacho Erondu, Nehemiah Adebayo, Micheal Olaolu Arowolo, Moses Kazeem Abiodun. "Different Encryption and Decryption Approaches for Securing Data."2022.

[11] Muhammad Bilal Qureshi, Muhammad Shuaib Qureshi , Saqib Tahir , Aamir Anwar, Saddam Hussain, Mueen Uddin and Chin-Ling Chen, Volume 14,Issue 4 ,28 March 2022.

[12] Sultan Almakdi, Brajendra Panda,Mohammed S. Alshehr "An Efficient Secure System for Fetching Data From the Outsourced Encrypted Databases.Volume 9" June 4, 2021.

[13] Chin-Chen Chang and Chao-Wen Chan, A database record encryption scheme using the RSA public key cryptosystem and its master keys,ICCNMC '03: Proceedings of the 2003 International Conference on Computer Networks and Mobile Computing (Washington, DC, USA), IEEE Computer Society,2003.

[14] Luc Bouganim and Philippe Pucheral, Chip-secured data access:confidential data on untrusted servers, VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases, VLDB Endowment, 2002, pp. 131–142.

[15] BalaIyer, Sharad Mehrotra2,Einar Mykletun,GeneTsudik, and Yonghua Wu, A Framework for Efficient Storage Security in RDBMS,Advances in Database Technology - EDBT 2004 Volume 2992 of the series Lecture Notes in Computer Science pp 147-164

[16] Tingjian Ge and S. Zdonik, Fast, secure encryption for exing in a column-oriented

DBMS, Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on, 2007, pp. 676–685.

[17] Berent, A. (2013). Advanced Encryption Standard by Example. Document available at URL http://www. networkdls. com/Articles/AESbyExample. pdf (April 1 2007) Accessed: June.

[18] Nadeem, H (2006). A performance comparison of data encryption algorithms," IEEE Information and Communication Technologies, (pp. 84- 89).

[19] Curtmola, R., Garay, J. A., Kamara, S., & Ostrovsky, R. (2006). Searchable symmetric encryption: Improved definitions and efficient constructions. In Proceedings of the 13th ACM conference on Computer and communications security (pp. 79-88). ACM. doi: 10.1145/1180405.1180418.

[20] Naveed, M., Kamara, S., & Wright, C. V. (2010). Inverted index for encrypted databases: beyond the cloud. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data (pp. 801-812). ACM.