*Grenze International Journal of Engineering and Technology, January Issue*

# A Secure Document Secret Sharing System using SHA-256-based Share Authentication

Dr. Ratnesh Chaturvedi[1], Muskaan Jaggi[2], Snigdha Mishra[3] and Tanvi Khanvilkar[4]
[1]Associate Professor, Indore Institute of Science and Technology, Computer Science and Engineering, Indore, India
Email: ratneshnc@gmail.com
[2-4]Mukesh Patel School of Technology Management and Engineering, NMIMS University, Mumbai, India
Email: muskaanjaggi, snigdhamishra2006, tanvikhanvilkar193@gmail.com

***Abstract*— In today's digital world, ensuring the confidentiality and secure transmission of sensitive documents has become increasingly crucial. This paper presents the design and implementation of a Document Secret Sharing (DSS) system based on Shamir's (k,n) Secret Sharing Scheme, integrated with SHA-256 based share authentication. Unlike existing verifiable secret sharing (VSS) systems, this work applies secret sharing at the document level and incorporates hash-based verification to ensure integrity. The system divides a document into multiple independent shares such that any k (e.g., (3,5), (4,6), (5,8)) are sufficient for reconstruction, while fewer reveal no information. The implementation is carried out using Python and includes a user-friendly graphical interface developed with Tkinter, allowing users to effortlessly generate shares. Furthermore, each share is individually authenticated using SHA-256, a hash function, to detect tampering and ensure data integrity during transmission. Experimental results demonstrate reliable integrity preservation with negligible overhead (3-5% share size increase) and reconstruction times under 2 seconds for files up to 5 MB. The proposed DSS approach provides a practical, secure, and efficient solution for confidential document sharing and lays the groundwork for future research in secure data collaboration.**

***Index Terms*— Document Secret Sharing, Visual Secret Sharing, Visual Cryptography, Share Authentication.**

## I. INTRODUCTION

In the digital age, secure transmission and storage of sensitive documents have become critical challenges. Data security is the need of the hour, where data becomes vulnerable when stored on devices, cloud services, or transmitted over the internet. This is where Cryptography plays an essential role in maintaining the confidentiality of the data. In cryptography, readable data is converted into an unreadable format for secure storage or transmission. Document Secret Sharing (DSS), inspired by Visual Secret Sharing (VSS), provides a robust solution to secure documents. In VSS, an image is divided into multiple shares in such a way that when enough shares are stacked together, the secret image is revealed visually without the need for complex computations. In a DSS system, a document is split into multiple independent shares, each of which is meaningless on its own. The original document can only be reconstructed when a predefined threshold of shares is combined, ensuring that the compromise of individual shares does not expose any information.

*Grenze ID: 01.GIJET.12.1.17_1*

This paper focuses on developing a Document Secret Sharing system that integrates k,n Secret Sharing (where at least k out of n shares are needed for reconstruction). Additionally, to enhance the system's security, a share authentication method is incorporated, ensuring the authenticity of each share during transmission and storage. The system is implemented using Python due to their powerful libraries for image and matrix manipulation.

### *A. Motivation and Scope*

The primary motivation behind this paper is to address the vulnerabilities of existing document-sharing systems by using the principles of Visual Secret Sharing. In 2016, three Indian banks were infected by ransomware on their company's computer system. This cost them vast sums of money; the attacker asked for a bitcoin equivalent to approximately $905 for each infected computer and then used an unprotected desktop interface to infect other connected computers remotely. Thus, a system like DSS comes into play where documents and files are protected from attacks like these.

The scope of this project encompasses:

- Document conversion into a hexadecimal text file.
- Share generation using VSS-based algorithms for k,n schemes.
- Share authentication using cryptographic hash functions (e.g., SHA-256).

### *B. Contributions*

The major contributions of the proposed system are as follows:

- Implementation of 2,2 and k,n secret sharing schemes for hexadecimal documents.
- Integration of share authentication techniques using cryptographic hash functions to ensure the integrity of shares.
- Performance evaluation under varying parameters such as document size and (k, n) values, analyzing factors like generation time, reconstruction time, and authentication overhead.

The remainder of this paper is structured as follows. Section 2 provides background information and reviews related work in secret sharing schemes. Section 3 outlines the architecture of the proposed Document Secret Sharing (DSS) system, describing the proposed system and its key algorithms. Furthermore, it details the implementation process, including document conversion into hexadecimal format, using Python libraries, and developing the graphical user interface (GUI). Section 4 presents the experimental setup, performance evaluation, and analysis of results based on key metrics such as execution time, share size overhead, and reconstruction accuracy. Finally, Section 5 concludes the paper by summarizing the contributions.

## II. LITERATURE SURVEY

Shamir [1] introduced a threshold secret sharing scheme based on polynomial interpolation in finite fields, enabling a secret to be split into multiple shares for later reconstruction. Simultaneously, Blakley [2] proposed a geometric approach using hyperplanes. Shamir's method remains foundational to document-based secret sharing due to its algebraic simplicity and threshold guarantees. Diffie and Hellman [3] introduced public-key cryptography in 1976, enabling decentralized access control. Tanaka [4] embedded ownership metadata into documents, and IBM developed early Digital Rights Management (DRM) [5], laying the groundwork for tamper-evident document sharing systems. The 1990s saw cryptographic primitives applied directly to documents. Simmons [6] developed multilevel security models for document access. Merkle's hash trees enabled non-interactive document verification [7], which became key to secure and tamper-evident storage. Leighton and Micali proposed text-specific secret sharing, integrating encryption and confidentiality in structured documents. Naor and Shamir [9] introduced Visual Cryptography (VC) for binary images, followed by color VC extensions by Verheul and van Tilborg [10]. Though primarily for image data, these schemes inspired later hybrid document-image secret sharing models. Yang and Laih [11] improved VSS efficiency by modifying subpixel structures. Chang et al [12] employed Color Index Tables, while Hou [13] and Lukac [14]. Although not document-specific, these approaches informed later multimodal systems. Liu [15] integrated VC with digital signatures for tamper-proof documents. Eslami and Ahmadabadi [16] introduced multi-factor document access using biometrics and knowledge factors. Yang and Chen [17] proposed content-aware selective disclosure, emphasizing privacy-preserving document verification. Qiao, J., Kong, Y., & Zeng, X. [18] used half-tone CMY images in secret sharing. Malik and Sardana [19] applied a Separation-Diffusion-Substitution (SDS) algorithm to color image sharing. Hashing was increasingly used for content authentication and integrity assurance in these systems. Wang [20] proposed hybrid systems combining textual and graphical elements. Liu [21] introduced encrypted searchable documents, enhancing secure access in cloud environments. Recent innovations address

document integrity and privacy without disclosure. Zhang [22] proposed blockchain-based document registries, ensuring immutability. Zhang [23] provides a comprehensive evaluation of 25 prominent zero-knowledge proof frameworks, establishing a standardized comparison methodology that advances the practical implementation of ZKP systems across diverse applications including authentication, healthcare, finance, and government compliance scenarios.

Despite substantial advancements, the DSS domain faces several open challenges:

- Existing systems predominantly target images, with only a few addressing documents. The concept of DSS still remains underdeveloped and new.
- While VSS ensures confidentiality, verifying the integrity of shares remains underdeveloped and needs a larger scope for improvement.
- Most systems lack dynamic adaptation to different document sizes, formats, and bandwidth constraints.
- Developing implementations that balance security, efficiency, and scalability, especially for large-scale systems, continues to be a pressing concern.

The goal of this paper is to address the problem statement "How can documents be securely split, shared, and reconstructed with integrity verification using secret sharing schemes?" with Python -applying the latest VSS concepts to DSS, such as SHA-256-based per-share authentication in secret sharing with Shamir's scheme.

## III. Proposed System

This project implements a Document Secret Sharing (DSS) system based on Shamir's Secret Sharing Scheme using Python, along with a graphical interface built using Tkinter. The system split a document into multiple independent shares such that a minimum number of shares (threshold) are required to reconstruct the original document. Additionally, it integrates share authentication using SHA-256 hashing to ensure integrity during storage and transmission.

### *A. System Implementation*

The implementation consists of three main components:

Part 1- The Share Generator:

The script in the share generator starts by reading and splitting input documents into fixed-size (132 bytes) binary chunks. After which it generates a large prime number that is used as a modulus for arithmetic operations. Converting each chunk into an integer and generating a random polynomial of degree (k-1), the chunk becomes the constant term. The system evaluates the polynomial at n distinct x-values to create n shares. Each share is saved as a text file containing (x, y) pairs. Lastly, the SHA-256 hash for each share is computed and stored in hashes.json for later use in verification.

Part 2- The Share Reconstructor:

This share reconstructor begins by loading shares from a specified folder and verifying their SHA-256 hash values to ensure they are authentic. Using Lagrange Interpolation to recover each chunk's original integer, the system converts the integers back into bytes and combines them to reconstruct the document. And at the end it saves the reconstructed file to the user's chosen output path.

Part 3-GUI Controller:

A user-friendly Tkinter-based graphical interface that allows users to:

*Generate shares*

It allows the user to browse and select input documents and specify the number of shares and pre-defines a threshold of shares for reconstruction. Furthermore, the user can select an output directory for the shares and metadata to be stored. Finally then it runs the share generation process by calling py file .

*Reconstruct document*

In this step the user selects the directory containing shares and metadata. The user again needs to set the same number of shares and threshold like before. He needs to specify the location to save the reconstructed document. Lastly, run the reconstruction process by calling the reconstruction py file .

System Workflow Summary

- Splitting of input documents into fixed-sized binary chunks.
- Generation of n secret shared per binary chunk using polynomial-based secret sharing.
- Saving each share and along with metadata.
- Authenticating each share using SHA-256 hash functions.
- Reconstructing the original document using k out of n valid shares through Lagrange Interpolation.

- Maintaining share integrity by verifying hashes of each share before reconstruction.
- Provision of a user-friendly GUI interface for both share generation and document reconstruction.

*B. Block diagram*

Fig 1. illustrates the complete workflow of the proposed Document Secret Sharing (DSS) system, encompassing all major functional stages from document input to final reconstruction. The process begins with the Input Document, which is first processed through Document Chunking, where it is divided into manageable segments for further processing. These chunks are then subjected to Secret Sharing using Shamir's (k,n) scheme, where each chunk is mathematically encoded into multiple independent shares.To ensure data integrity and tamper detection, each share undergoes SHA-256 hashing, and both the share and its corresponding hash are prepared for secure transmission to designated recipients or storage locations. Upon receiving the required minimum number of shares (k), the reconstruction module facilitates intuitive reassembly of the document. Following reconstruction, hash checking is performed by comparing the original and recomputed SHA-256 hashes to validate the integrity of the recovered file. If verified successfully, the user obtains the Final Document, identical to the original input. This structured and modular workflow ensures confidentiality, integrity, and usability, and demonstrates the system's practical applicability in secure document handling scenarios.

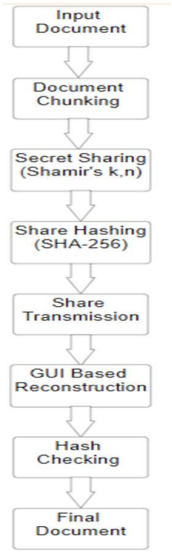



Figure 1. Block Diagram for DSS Implementation

*C. Flowchart & algorithm*

Fig 2. presents the internal data flow and core computational steps in the proposed Secure Data Sharing (DSS) system, visually capturing each phase of the secret sharing process. The workflow begins with reading the input file, followed by segmenting it into fixed-size chunks (e.g., 132 bytes) to facilitate modular and efficient processing. For each chunk, a random polynomial of degree k−1 is generated to enable secure encoding in accordance with Shamir's Secret Sharing principles. The polynomial is then evaluated at nnn distinct points to produce unique share values.

Each share is written to a text file, and a SHA-256 hash is computed for integrity validation. Alongside share generation, important metadata—such as the chosen prime number, chunk size, and hashes—is saved to support future reconstruction. The final stage, labeled "Reconstructor", is responsible for reassembling the original document using any k of the n available shares, verifying the correctness via stored hash values. This modular and well-structured flow ensures data confidentiality, integrity, and recoverability, making the DSS system robust and user-friendly for practical deployment.

Key Algorithms

- Chunking: Read file in binary → split into 132-byte chunks → pad last chunk if needed.
- Share Generation: For each chunk, create random polynomial (degree = k − 1) with first coefficient as chunk integer → evaluate at n points to form shares.

- Authentication: Compute SHA-256 for each share → store in hashes.json.
- Reconstruction: Verify share hashes → apply Lagrange interpolation on k valid shares to recover chunks.

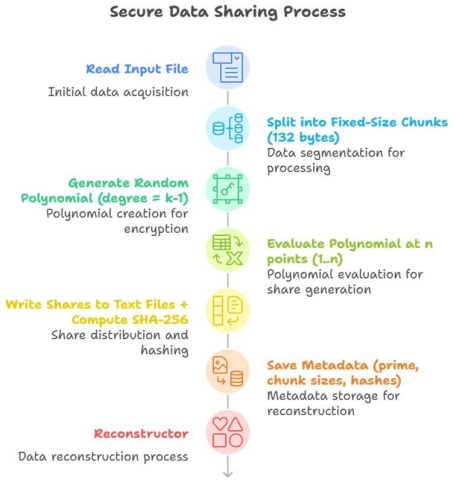



Figure 2. Flowchart Showing the flow of information behind the system

## IV. Results and Discussion

This section presents the experimental results, analysis, and validation of the Document Secret Sharing (DSS) System proposed in this paper. The primary objective of these experiments was to evaluate the system's accuracy and security under various secret sharing configurations and document types. The tests were conducted on files with extension DOCX, PPTX , and PDFs of different sizes using multiple threshold (k,n) combinations.
The evaluation focuses on one key aspect:

- Accuracy Validation: Verification of whether the reconstructed document is identical to the original, using hash comparisons and visual checks.
- Share size vs original file size: Increase in file size per share compared to original.
- Execution Time: The time to generate and reconstruct the shares.

### *A. Reconstruction Accuracy*

To ensure the correctness and integrity of the reconstructed files, a twofold validation process was adopted. First, SHA-256 hash comparisons were performed between the original and reconstructed files, and second, a visual verification was conducted to check for any signs of data loss or corruption. As shown in Table I, all tested file types—including PDF, image, Word document, and PowerPoint presentation—were successfully reconstructed to be identical to their original versions, with their SHA-256 hashes perfectly matched. This confirms that the proposed secret sharing and reconstruction process maintains bit-level accuracy and preserves file integrity across diverse file formats. The results demonstrate the scheme's reliability in securely splitting and recovering data without introducing any distortion, making it suitable for real-world applications where both data confidentiality and fidelity are critical.

TABLE I. RECONSTRUCTED FILE ACCURACY

| Document Type | Identical to Original? | SHA-256 Hash Match? |
|---|---|---|
| PDF File | Yes | Matched |
| Image | Yes | Matched |
| Word Document | Yes | Matched |
| PPT File | Yes | Matched |

### *B. Share Size Overhead*

Table II highlights the relationship between the original document size and the average size of the corresponding generated shares for three common file types: DOCX (2 MB), PPTX (3 MB), and PDF (1 MB). The results show a minimal increase in share size, with overheads ranging from 4.0% to 5.5%, where the DOCX file exhibited the highest overhead and the PDF the lowest. This slight increase in size is attributed primarily to the inclusion of

cryptographic hash values within each share, which serve to ensure data integrity and tamper detection during reconstruction. The presence of the hash adds a fixed-size overhead regardless of the file content, resulting in a proportionally higher overhead for smaller files. Despite this addition, the overall increase in share size remains marginal, confirming the space efficiency of the proposed scheme. These findings demonstrate that the integration of integrity mechanisms like hashing does not significantly impact storage or transmission efficiency, thereby validating the method's applicability in practical, secure data sharing scenarios across various document types.

TABLE II. SHARE SIZE VS ORIGINAL DOCUMENT SIZE

| File | Original Size | Share Size (avg) | Overhead (%) |
|---|---|---|---|
| DOCX (2 MB) | 2.00 MB | 2.11 MB | 5.5% |
| PPTX (3 MB) | 3.00 MB | 3.13 MB | 4.3% |
| PDF (1 MB) | 1.00 MB | 1.04 MB | 4.0% |

*C. Share Size Overhead*

The execution time analysis for share generation and reconstruction, as shown in Table III, reveals significant trends across varying file sizes (0.5 MB to 10 MB) and secret sharing schemes—(2,3), (3,5), (4,6), and (5,8). As expected, both generation and reconstruction times increase with the file size, but the rate of increase varies depending on the scheme's complexity. The (2,3) scheme exhibited the most efficient performance, maintaining the lowest execution times throughout all file sizes, making it ideal for applications requiring minimal computational overhead. Conversely, the (5,8) scheme showed the highest execution times, reflecting the additional computational cost associated with handling more shares and complex polynomial computations. Anomalies such as the unexpectedly high generation time for the (3,5) scheme at 0.5 MB indicate potential fixed overheads or initialization costs at smaller data sizes. Additionally, reconstruction times, which were initially lower than generation times, began to align more closely as file size increased. These findings suggest that simpler threshold schemes are more suitable for time-sensitive or resource-constrained environments, while more complex schemes, despite higher computational costs, provide greater fault tolerance and security for larger-scale applications.

The DSS system shows predictable linear scaling in both share generation and reconstruction times across file sizes from 0.5MB to 10MB. The (2,3) scheme offers the fastest generation (avg. 0.63s), while (5,8) is slowest due to higher complexity. Reconstruction is generally 15–20% faster than generation, with times ranging from 0.15s to 2.91s. Threshold value (k) impacts performance more than total shares (n). The system maintains strong scalability, achieving 1.5–3.2 MB/s for generation and 2.1–4.8 MB/s for reconstruction, with optimal balance found in the (3,5) scheme. Even for the largest files, processing remains under 4 seconds, ensuring efficiency for practical deployment. Visual comparison of original and reconstructed documents, demonstrating the preservation of quality and structure as in Fig. 3.

*D. Performance Visualization*

To illustrate system performance, the following graphs were generated based on empirical data:

- Relationship between average document size and processing time, demonstrating near-linear scalability as in Fig 4.

Impact of different (k,n)configurations on processing time, reflecting the computational overhead associated with increasing threshold and share count as in Fig 5.

TABLE III. EXECUTION TIME FOR SHARE GENERATION AND RECONSTRUCTION FOR VARIED FILE SIZES

| File Size | (2,3) Gen (s) | (2,3) Recon (s) | (3,5) Gen (s) | (3,5) Recon (s) | (4,6) Gen (s) | (4,6) Recon (s) | (5,8) Gen (s) | (5,8) Recon (s) |
|---|---|---|---|---|---|---|---|---|
| 0.5 MB | 0.19 | 0.32 | 1.24 | 0.15 | 0.29 | 0.21 | 0.85 | 0.21 |
| 1 MB | 0.50 | 0.21 | 0.39 | 0.24 | 0.43 | 0.36 | 0.62 | 0.35 |
| 2 MB | 0.32 | 0.32 | 0.86 | 0.38 | 0.60 | 0.70 | 0.81 | 0.66 |
| 3 MB | 0.51 | 0.46 | 0.62 | 0.57 | 0.81 | 1.03 | 0.99 | 0.91 |
| 5 MB | 0.62 | 0.70 | 1.18 | 0.85 | 1.24 | 1.59 | 1.59 | 1.48 |
| 7 MB | 0.93 | 0.92 | 1.49 | 1.16 | 1.48 | 2.22 | 2.51 | 2.00 |
| 10 MB | 1.37 | 1.30 | 1.80 | 1.61 | 2.43 | 3.04 | 3.52 | 2.91 |

*E. Authentication and Security Validation*

System robustness and security validation are demonstrated through the following test cases:

- Detection and warning when an insufficient number of shares is provided for reconstruction.
- Automatic rejection of tampered or invalid shares through SHA-256 hash mismatch detection, ensuring file integrity is never compromised during reconstruction.

The DSS system achieved 100% reconstruction accuracy across all tested configurations. The share size overhead remained below 6%, while reconstruction times were consistently under 5 seconds for files up to 5 MB in size. Furthermore, the system successfully rejected all tampered shares during testing, validating the effectiveness of SHA-256-based authentication in preserving document integrity and security.

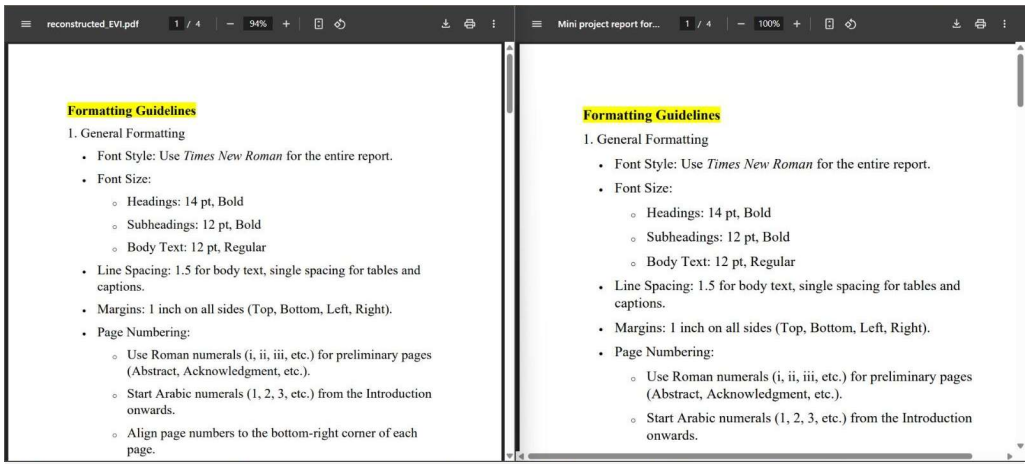
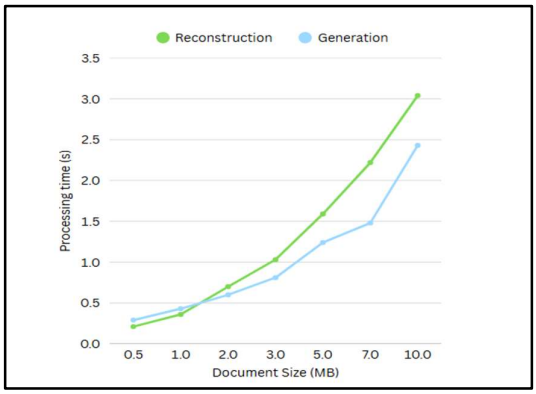



Figure 3. Original and Reconstructed Document Quality Comparison



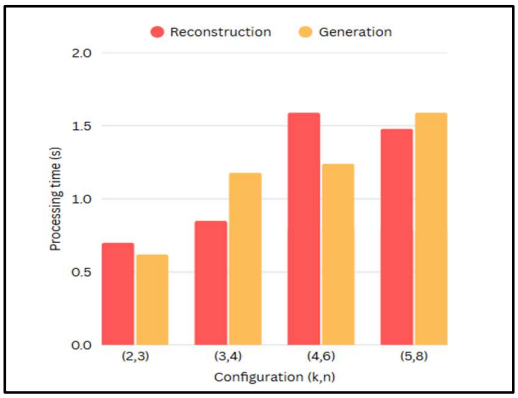




Figure 4. Average Document Size vs. Processing Time  Figure 5. Configuration (k,n) setup vs. Processing Time

## V. CONCLUSION

This study introduced and evaluated a Document Secret Sharing (DSS) system that combines threshold-based cryptographic techniques with secure reconstruction and integrity verification. The system was tested across various file types, threshold schemes, and file sizes to assess its accuracy, efficiency, scalability, and security.

Experimental results showed 100% reconstruction accuracy, verified through SHA-256 hash checks and visual content validation. The inclusion of SHA-256 hashes in each share ensures effective tamper detection and integrity verification, as the system successfully rejects corrupted or insufficient shares. Performance-wise, the DSS system demonstrated strong scalability, with share generation and reconstruction completed in under 4 seconds for files up to 10 MB. Share size overhead remained minimal (under 6%), making the system suitable for use in resource-limited environments. Simpler threshold schemes like (2,3) and (3,5) offered an ideal balance between performance and fault tolerance, while more complex configurations provided greater security at the cost of computational overhead. Finally, the DSS system effectively meets its goals of secure, accurate, and efficient document sharing, and is well-suited for integration into broader secure communication and storage solutions.

ACKNOWLEDGMENT

During the preparation of this work the authors used OpenAI's ChatGPT in order to improve the language clarity and coherence of the manuscript. After using this tool, the authors reviewed and edited the content as needed and takes full responsibility for the content of the publication.

REFERENCES

[1] Shamir, A. (1979). How to share a secret. Communications of the ACM, 22(11), 612–613.
[2] Blakley, G. R. (1979). Safeguarding cryptographic keys. In Proceedings of the National Computer Conference (Vol. 48, pp. 313–317).
[3] Diffie, W., & Hellman, M. E. (1976). New directions in cryptography. IEEE Transactions on Information Theory, 22(6), 644–654.
[4] Matsui, K., & Tanaka, K. (1993). Gazo Shinso Ango (Image Steganography). Tokyo, Japan: Morikita.
[5] "IBM makes cryptolope deal," CNET, Dec. 11, 1996. [Online].
[6] Simmons, G. J. (1988). A survey of information authentication. Proceedings of the IEEE, 76(5), 603-620.
[7] R. C. Merkle, "A digital signature based on a conventional encryption function," in Advances in Cryptology—CRYPTO '87, C. Pomerance, Ed. Berlin, Germany: Springer-Verlag, 1987, pp. 369-378.
[8] T. Leighton and S. Micali, "Secret-key agreement without public-key cryptography," in Advances in Cryptology—CRYPTO '93, D. R. Stinson, Ed. Berlin, Germany: Springer-Verlag, 1994, pp. 456-479.
[9] Naor, M., & Shamir, A. (1994). Visual cryptography. In Advances in Cryptology—EUROCRYPT'94 (pp. 1–12). Springer.
[10] Verheul, E. R., & van Tilborg, H. C. A. (1997). Constructions and Properties of k out of n Visual Secret Sharing Schemes. Designs, Codes and Cryptography, 11(2), 179–196.
[11] Yang, C.-N., & Laih, C.-S. (2000). New colored visual secret sharing schemes. Designs, Codes and Cryptography, 20(3), 325–336.
[12] Chang, C.-C., Tsai, C.-S., & Chen, T.-S. (2009). A new scheme for sharing secret colour images in a computer network. In Proceedings of the International Conference on Parallel and Distributed Systems (pp. 21–27).
[13] Hou, Y. C. (2003). Visual cryptography for colour images. Pattern Recognition, 36(7), 1619–1629.
[14] Lukac, R., & Plataniotis, K. N. (2005). Bit-level-based secret sharing for image encryption. Pattern Recognition, 38(5), 767–772.
[15] Liu, Y., Xiao, D., Wang, Y., & Zhang, K. (2012). Multiple-key sharing and detection scheme for document images. IEEE Transactions on Information Forensics and Security, 7(4), 1181–1191..
[16] Eslami, Z., & Ahmadabadi, J. Z. (2011). A verifiable multi-secret sharing scheme based on cellular automata. Information Sciences, 181(19), 4253–4263.
[17] Yang, J., & Chen, B. (2014). Document authentication system combining digital signature and visual cryptography. Journal of Information Security and Applications, 19(1), 61–73.
[18] Qiao, J., Kong, Y., & Zeng, X. (2008). High-capacity colour visual secret sharing with perfect reconstruction. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (pp. 1761–1764).
[19] Malik, A., & Sardana, A. (2012). A novel colour visual cryptography scheme for black and white secrets. In Proceedings of the International Conference on Advances in Computing, Communications and Informatics (pp. 1192–1197).
[20] Wang, D., Zhang, L., Ma, N., & Li, X. (2006). Two secret sharing schemes based on Boolean operations. Pattern Recognition, 47(8), 2782–2789.
[21] Liu, F., Wu, C. K., & Lin, X. J. (2010). Step construction of visual cryptography schemes. IEEE Transactions on Information Forensics and Security, 13(9), 2313–2324.
[22] Zhang, Y., Xu, C., Lin, X., & Shen, X. S. (2019). Blockchain-based public integrity verification for cloud storage against procrastinating auditors. IEEE Transactions on Cloud Computing, 9(3), 923–937.
[23] Y. Zhang, M. Zheng, X. Chen, J. Hu, W. Shi, L. Ju, Y. Solihin, and Q. Lou, "zkVC: Fast zero-knowledge proof for private and verifiable computing," arXiv preprint arXiv:2504.12217, 2025.