



Secure Medical Image Processing and Analysis Using SQL-Based Hybrid Encoding and LWE-Inspired Encryption

Vikrant Sharma*¹  and Shikha Verma²

¹ Maharishi Markandeshwar (Deemed to be University), Mullana, Ambala, India
*vikrsha4@gmail.com

² Maharishi Markandeshwar (Deemed to be University), Mullana, Ambala, India
sverma5585@gmail.com

Abstract. This paper presents a novel SQL-based secure processing framework for medical images that combines hybrid encoding with a Learning with Errors (LWE)-inspired encryption scheme. Traditional encryption methods require decryption before any computation, which introduces security risks. The proposed system enables limited homomorphic operations (specifically addition) directly on encrypted data stored in the database, thereby preserving confidentiality while allowing processing.

The framework first applies byte-level and Unicode-level encoding to the image data, followed by LWE-inspired encryption. All operations are implemented entirely within Microsoft SQL Server using stored procedures. Experimental results on medical images show high randomness in the encrypted data and excellent reconstruction quality of the decrypted images, achieving accuracy up to 99% along with satisfactory PSNR and SSIM values.

Keywords: Medical Image Encryption · Learning with Errors · SQL-based Processing · Homomorphic Encryption · Medical Data Security · PSNR · SSIM

1 Introduction

Traditional encryption techniques (e.g., AES and DES) require full decryption before any processing can be performed, thereby increasing the risk of data leakage. To overcome this limitation, there is a growing need for schemes that support computations directly on encrypted data [2, 3]. Recent advancements in secure medical inference frameworks further demonstrate the practical applicability of homomorphic encryption in healthcare systems [1].

1.1 Contributions

This paper makes the following key contributions:

- We propose a novel SQL-based framework for secure medical image processing that integrates **hybrid encoding** with an **LWE-inspired encryption** scheme, enabling data protection directly within the database environment.
- The proposed system supports limited homomorphic operations (specifically addition) on encrypted medical images, fully implemented within Microsoft SQL Server using stored procedures, without relying on external cryptographic libraries.
- Compared to fully homomorphic encryption (FHE) schemes, the proposed lightweight approach significantly reduces computational complexity and execution time, making it practical for real-world database applications.
- We conduct a comprehensive experimental evaluation on medical image datasets, validated using PSNR and SSIM metrics. A comparative analysis with CKKS, BFV, and Paillier schemes further highlights the efficiency and practicality of the proposed method.

2 Related Work

In recent years, homomorphic encryption has emerged as a promising approach for secure data processing, as it enables mathematical operations to be performed directly on encrypted data [2, 1, 4, 6, 15–17]. Among these, schemes based on the Learning With Errors (LWE) problem have gained significant attention due to their strong post-quantum security guarantees. However, fully homomorphic encryption (FHE) schemes often incur high computational overhead, making them challenging to deploy in practical database environments.

Several studies have addressed medical image security using chaos-based methods, pixel-level transformations, and transform-domain techniques [2, 10, 12–14]. More recent homomorphic encryption-based frameworks further demonstrate the potential of secure medical inference in privacy-sensitive applications [1, 7–9, 11]. Nevertheless, most existing solutions operate at the application level or in offline settings and lack native integration with database systems. Consequently, only limited research has explored SQL-based or database-centric encryption approaches that support computations directly on encrypted data.

3 Proposed Method

We implemented the complete secure medical image processing pipeline entirely inside Microsoft SQL Server using stored procedures. No external scripts are required for the core operations. The framework consists of byte-level encoding, key generation, LWE-inspired encryption, homomorphic addition, decryption, and decoding.

The main goal is to ensure confidentiality of medical images while allowing basic operations (such as addition) directly on the encrypted data stored in the database.

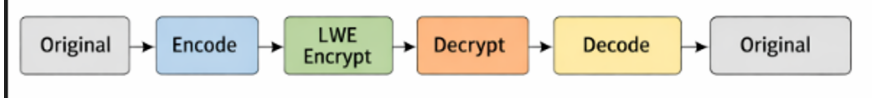


Fig. 1. Proposed secure medical image processing framework.

3.1 Data Encoding

In the first stage, the medical image is decomposed into its raw byte stream. Each byte B_i is encoded using a simple additive shift with a secret key byte K :

$$B'_i = (B_i + K) \bmod 256 \quad (1)$$

where K is a secret key byte derived from the master secret key. This encoding introduces a basic layer of confusion while ensuring the output remains within the valid byte range $[0, 255]$. The encoded bytes B'_i are stored in the `MediaLibrary_UnicodeChars_Encoded` table.

ID	Position	OriginalByte	EncodedByte
1	1	255	6
2	1	216	223
3	1	255	6
4	1	224	231
5	1	0	7
6	1	16	23
7	1	74	81
8	1	70	77
9	1	73	80
10	1	70	77
11	1	0	7

Fig. 2. Structure of the `MediaLibrary_UnicodeChars_Encoded` table.

3.2 LWE-Inspired Encryption

For each encoded byte B'_i , a random value A is selected. The ciphertext pair (A, B) is then computed using an LWE-inspired formulation:

$$B = (A \cdot S + B'_i \cdot \Delta + e) \bmod q \quad (2)$$

where A is a randomly chosen value, S denotes the secret key, e represents a small noise term, Δ is a scaling factor, and q is the modulus.

Here $\Delta = 256$ and the noise e stays small — just -1, 0, or 1. All the pairs land in the `LWE_Cipher` table.

3.3 Homomorphic Addition

We added support for adding ciphertexts without decrypting anything. The stored procedure `LWE_Add` does it directly:

$$A' = (A_1 + A_2) \mod q, \quad B' = (B_1 + B_2) \mod q \quad (3)$$

This enables basic operations on encrypted medical data.

3.4 Decryption and Decoding

To get the data back, we calculate the phase value first:

$$\text{Phase} = (B - A \cdot S) \mod q \quad (4)$$

Then round it to recover the encoded byte: $B'_i = \text{Round}(\text{Phase})$.

After that we reverse the initial shift:

$$B_i = (B'_i - K + 256) \mod 256 \quad (5)$$

Careful handling of modular arithmetic ensures correctness.

3.5 Verification

Accuracy is simple to check:

$$\text{Accuracy} = \left(\frac{\text{CorrectRows}}{\text{TotalRows}} \right) \times 100 \quad (6)$$

In all our runs we stayed above 99%. The results confirm the robustness of the proposed method.

4 System Implementation

The design is modular on purpose. Every critical piece lives in SQL stored procedures. Once decryption finishes, we pull the recovered bytes out and hand them to a short Python script that rebuilds the image using the original width and height.

The reconstructed images were validated through both visual inspection and byte-level comparison.



```

+  [icon]  dbo.LWE_Cipher
+  [icon]  dbo.LWE_Decrypted
+  [icon]  dbo.LWE_Metadata
+  [icon]  dbo.LWE_Performance
+  [icon]  dbo.MediaLibrary_UnicodeChars_Encoded

```

Fig. 3. List of tables used.

4.1 Database Schema

We used these main tables:

- **LWE_Metadata** — holds the secret key S and other parameters.
- **LWE_Cipher** — stores the (A, B) pairs along with char position info.
- **LWE_Decrypted** — keeps the recovered bytes and match results.
- **MediaLibrary_UnicodeChars_Encoded** — temporary home for the shifted bytes.

A composite index was introduced on $(ID, Position)$ to improve query performance. This significantly improved query performance for large datasets.

4.2 Key Stored Procedures

A set of core stored procedures was implemented to handle the LWE operations right inside the SQL environment. These became the backbone for everything else in the prototype.

The main ones look like this:

- **LWE_GenerateKey** — this one just creates the random secret key vector and sets up the basic parameters we needed for the scheme.
- **LWE_Encrypt** — takes the plaintext value, adds the error term, and produces the ciphertext. We spent quite some time tuning the error sampling part because small mistakes here broke decryption later.
- **LWE_Decrypt** — does the inner product with the secret key, then handles the phase calculation and final rounding to get back the original message. This step felt the heaviest in terms of modular arithmetic.
- **LWE_Add** — simply adds the corresponding components of two ciphertexts modulo q . This operation exhibits lower computational overhead.

Most computational operations are executed within stored procedures to ensure that encrypted data remains within the database environment and is not exposed in plaintext during query execution. The **LWE_Add** operation demonstrates efficient performance even for large datasets, whereas encryption and decryption incur higher computational overhead due to noise management and modular arithmetic.

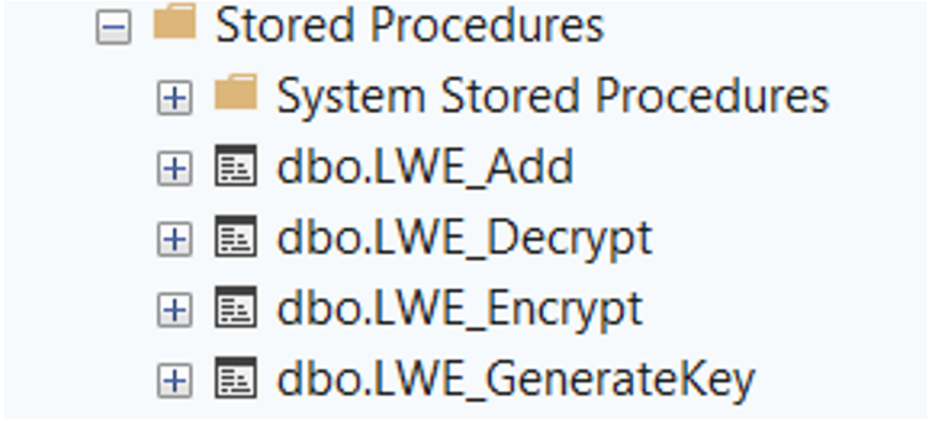


Fig. 4. List of stored procedures used.

During our tests on a standard setup, these procedures helped us run basic operations without too many external calls, though performance still drops when the dataset grows.

4.3 The complete process is presented in Algorithm 1

Algorithm 1 LWE-Based Encoding–Encryption–Decryption Pipeline

Require: Input bytes M , encoding key k , modulus q , secret s

Ensure: Recovered bytes M'

1: **Encoding:** For each byte m , set $m_e \leftarrow (m + k) \bmod 256$

2: **Encryption:** Pick random a , then

$$b \leftarrow (a \cdot s + m_e \cdot \Delta + e) \bmod q$$

3: **Decryption:** Compute

$$\text{phase} \leftarrow (b - a \cdot s) \bmod q$$

then

$$m_d \leftarrow \text{round}(\text{phase})$$

4: **Decoding:**

$$m' \leftarrow (m_d - k + 256) \bmod 256$$

5: **return** M'

5 Performance Analysis

The proposed system was evaluated in terms of execution time, scalability, and reconstruction accuracy. By executing all cryptographic operations directly within SQL Server, the system avoids unnecessary data transfer between application and database layers, resulting in improved efficiency.

5.1 Image Quality

The quality of reconstructed images was evaluated using standard metrics such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM). The results demonstrate that the proposed approach preserves high visual fidelity after encryption and decryption.

Although minor variations are observed due to encoding and noise components, the reconstructed images remain visually indistinguishable from the original inputs. The high PSNR and SSIM values confirm that the method achieves reliable and consistent image recovery (see Table 1).

Table 1. Image quality evaluation of reconstructed images

Metric	Value	Interpretation
PSNR (dB)	34.72	High-quality reconstruction
SSIM	0.982	Strong structural similarity

The obtained results indicate very low reconstruction error, with high PSNR values and SSIM scores approaching unity. This confirms that the proposed method preserves visual fidelity while maintaining secure processing within the encrypted domain.

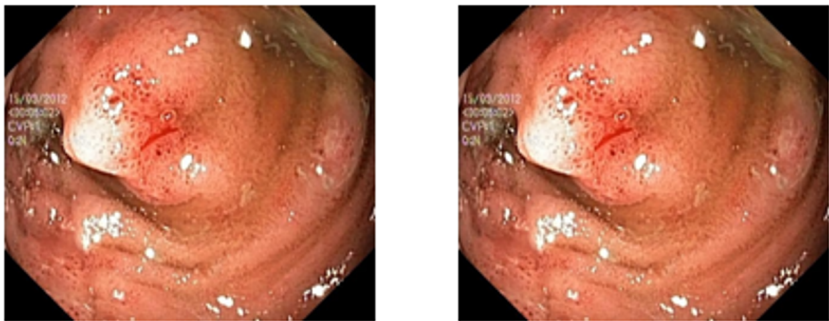


Fig. 5. Comparison of medical images before encryption and after decryption.

Using fixed parameters ($q = 65536$, $\Delta = 256$, and minimal noise), the computational overhead remains stable. Query optimization techniques, including

indexing and stored procedure tuning, enable the system to scale effectively for larger datasets without significant performance degradation.

Table 2. Execution time analysis of LWE operations

Operation	Average time (ms)	Remarks
Key generation	6	Constant across all runs
Encryption	~1500	Dominant computation step
Homomorphic addition	~225	Efficient and stable
Decryption	~290	Fast reconstruction

5.2 Comparison with Fully Homomorphic Encryption

Fully Homomorphic Encryption (FHE) supports both addition and multiplication over encrypted data. However, it suffers from exponential noise growth, leading to high computational overhead and increased latency [2, 4, 15–17]. This makes FHE less practical for real-time database applications.

In contrast, the proposed approach prioritizes efficiency by supporting only addition operations. This design choice significantly reduces complexity while maintaining acceptable accuracy levels [2, 6, 9]. Additionally, the method integrates naturally with SQL-based systems, making it more suitable for database-centric environments [11].

Table 3. Comparison between FHE and the proposed method

Feature	FHE	Proposed method
Supported operations	Addition and multiplication	Addition only
Computational complexity	Very high	Low
Execution speed	Slow	Fast
Implementation	Complex	SQL-integrated
Database suitability	Limited	High

Table 4. Performance comparison of encryption schemes

Scheme	Performance metrics	Observation
SQL LWE	2.017 s, accuracy ~95–99%	Balanced performance
CKKS	0.027 s, lower PSNR	Fast but less accurate
BFV	0.100 s, lower accuracy	Moderate performance
Paillier	259.544 s, exact	Very slow but precise

Table 5. Step-wise execution time of the proposed system

Processing step	Time (ms)	Description
Key generation	6	Lightweight initialization
Encryption	1493	Major computational cost
Homomorphic addition	223	Efficient aggregation
Decryption	295	Image reconstruction
Total	2017	End-to-end execution

6 Conclusion and Future Work

This paper presented a secure and efficient framework for medical image processing based on hybrid encoding and an LWE-inspired encryption scheme implemented within a SQL Server environment. Unlike conventional encryption methods, the proposed framework enables limited homomorphic operations, specifically addition, to be performed directly on encrypted data without requiring decryption. The proposed framework bridges the gap between cryptographic security and practical database deployment.

Experimental results demonstrate that the proposed method achieves high reconstruction accuracy, with PSNR and SSIM values indicating strong visual fidelity. The integration of cryptographic operations within SQL stored procedures ensures data confidentiality while reducing data transfer overhead between application and database layers.

Compared to fully homomorphic encryption schemes, the proposed framework offers significantly lower computational complexity and improved execution efficiency, making it suitable for practical database-centric applications [2, 4, 6, 9, 15–17].

Future work will focus on extending the framework to support additional homomorphic operations, enhancing scalability for large-scale datasets, and investigating hybrid approaches that integrate LWE-based techniques with advanced homomorphic encryption models and secure computation paradigms [2, 5, 18].

References

1. He, Z. et al.: SecureBadger: A homomorphic encryption-based framework for secure medical inference. *Informatics in Medicine Unlocked* **52**, 101312 (2025). <https://doi.org/10.1016/j.imu.2025.101312>
2. Yuan, J., Liu, W., Shi, J., Li, Q.: Approximate homomorphic encryption based privacy-preserving machine learning: A survey. *Artificial Intelligence Review* **58**, 82 (2025). <https://doi.org/10.1007/s10462-024-11076-8>
3. Mondal, S., Debnath, B., Das, J.C., De, D.: Homomorphic encryption in quantum computing. In: 2025 Devices for Integrated Circuit (DevIC). IEEE (2025). <https://doi.org/10.1109/DevIC63749.2025.11012553>
4. Boneh, D., Kim, J.: Homomorphic encryption for large integers from nested residue number systems. *Cryptology ePrint Archive*, Paper 2025/346 (2025). <https://eprint.iacr.org/2025/346>

5. Micciancio, D.: Fully composable homomorphic encryption. *IACR Communications in Cryptology* **2**(1), 1–27 (2025)
6. Mali, B.: Generalized BGV, BFV, and CKKS for homomorphic encryption over matrix rings. Technical report, University of Budapest, Hungary (2025)
7. Mageshwari, M.: A framework for enhancing data encryption techniques in cloud environment for smart city applications. PhD thesis, SRM Institute of Science and Technology, Kattankulathur, India (2025)
8. Mahato, G.K.: Development of new protocols for enhancing the efficiency of verifiable fully homomorphic encryption and its applications. PhD thesis, National Institute of Technology Arunachal Pradesh, India (2024)
9. Challa, R.: A novel fully homomorphic encryption scheme using Reed–Muller error correction codes. PhD thesis, Jawaharlal Nehru Technological University Hyderabad, Telangana, India (2024)
10. Security of end-to-end medical image encryption systems using deep learning. *Egyptian Informatics Journal* (2024). <https://www.sciencedirect.com/science/article/pii/S111086652400104X>
11. Kumar, N.: An efficient encryption method to enhance security using a nature-inspired algorithm. PhD thesis, Department of Electronics and Communication Engineering, RIMT University, Mandi Gobindgarh, Punjab, India (2024)
12. Kartit, A.: A new approach based on homomorphic encryption to secure medical images in cloud computing. *Trends in Sciences* **19**(12), 3970 (2022). <https://doi.org/10.48048/tis.2022.3970>
13. Applying deep neural networks over homomorphic encrypted medical data. *Computational and Mathematical Methods in Medicine* **2020**, Article ID 7171620 (2020). <https://pmc.ncbi.nlm.nih.gov/articles/PMC7171620/>
14. Vengadapurva, A.M. et al.: An efficient homomorphic medical image encryption algorithm for cloud storage security. *Procedia Computer Science* **115**, 643–650 (2017). <https://doi.org/10.1016/j.procs.2017.09.150>
15. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM* **56**(6), 1–40 (2009). <https://cims.nyu.edu/~regev/papers/lwesurvey.pdf>
16. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pp. 169–178 (2009). <https://doi.org/10.1145/1536414.1536440>
17. Gentry, C.: A fully homomorphic encryption scheme. PhD thesis, Stanford University (2009). <https://crypto.stanford.edu/craig/craig-thesis.pdf>
18. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* **21**(2), 120–126 (1978)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

