Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 29 (2026), No. 10-C, pp. 4787–4794
DOI : 10.47974/JDMSC-3092

Adaptive semantic guardrails : Dynamic magic square obfuscation as a positional cipher for NL2SQL environments

Rahul Kumar Sharma *
School of Computer Science & Information Technology (SCSIT)
Devi Ahilya Vishwavidyalaya
Indore 452001
Madhya Pradesh
India

Vivek Kapoor †
Department of Information Technology
Institute of Engineering & Technology
Devi Ahilya Vishwavidyalaya
Indore 452017
Madhya Pradesh
India

Abstract

Natural Language to SQL (NL2SQL) interfaces has made data far more accessible, but they have also opened up a new kind of weakness we call "Semantic Drift." Because these systems are probabilistic, a Large Language Model (LLM) can produce SQL that is syntactically correct yet still breaks implicit privacy rules through hallucination or adversarial prompt injection. Conventional, deterministic security models are not built for this: they check identity, not semantic intent. To close that gap, we apply discrete combinatorial mathematics—specifically, dynamically generated Magic Squares—to build a positional symmetric cipher that functions as an adaptive Moving Target Defense (MTD). A Tri-State Decision Engine sits inside transparent JDBC middleware, balancing data integrity against active defense. When a query looks anomalous, the system responds with high-entropy obfuscation generated through the Siamese method, scrambling sensitive data on the fly and defeating frequency analysis in the process. In our evaluation, this approach caught 97.4% of adversarial injections while adding only 6.8ms of latency overhead, pointing toward a new cryptographic standard for zero-trust database connectivity in AI-driven systems.

Subject Classification: *Primary 94A60, Secondary 05B15, 68P24.*

Keywords: *Cryptography, Magic squares, Moving target defense, Positional cipher, Information entropy, NL2SQL, Semantic drift.*

* *E-mail:* **rahulsharma21@gmail.com (Corresponding Author)**
† *E-mail:* **vkapoor@ietdavv.edu.in**

© Taru Publications

1. Introduction

When generative AI meets enterprise databases, a real paradox shows up: Large Language Models (LLMs) are probabilistic and stochastic by nature, while relational algebra and the database engines built on it are strictly deterministic [8]. That mismatch is exactly where standard cryptographic and access-control boundaries start to fail [8]. Before LLMs, a SQL query was a deterministic construct and security was essentially binary—allowed or denied. Now, the query itself is a stochastic output that can "hallucinate" connections between data domains that were never meant to intersect. And because the LLM agent typically connects through an authenticated service account, the database engine has no reason to distrust this "Drifted Query"—it simply executes it.

Our work builds on earlier frameworks for access control and relational integrity verification [1]-[3], and it draws on more recent research showing how useful analytical modeling can be for securing dynamic stream-processing systems [12]. Traditional security boundaries tend to struggle once inputs become probabilistic, so we adapt classical cryptographic matrix transformations [10] to protect the modern data layer. Combining these properties with a discrete combinatorial key-distribution scheme [11] lets us map out, fairly precisely, how far mathematical data obfuscation can be pushed.

This paper tackles that Stochastic-Deterministic Gap by moving cryptographic defense away from static storage and into the dynamic connectivity layer instead. As Figure 1 illustrates, we describe a cognitive firewall built around a new cryptographic use of Magic Squares. Drawing on the combinatorial properties of these matrices, we introduce a positional symmetric cipher that mathematically guarantees data obfuscation—effectively an active Moving Target Defense (MTD) [6] whenever semantic intent strays from what was authorized.

Here is how the rest of the paper unfolds. Section 2 formalizes the mathematical threat landscape of NL2SQL interfaces, mapping out semantic drift vectors and analyzing systemic vulnerabilities through a full STRIDE framework. Section 3 lays out our mathematical methodology, covering how we derive unsupervised anomaly scores with Isolation Forests and the discrete combinatorial mechanics we use to scale positional entropy. Section 4 describes the real-time execution architecture and decision logic, including the tri-state cryptographic threshold parameters and the dynamic Siamese method used for runtime key generation. Section 5 walks through an adversarial data-exfiltration scenario and provides mathematical proof of frequency-analysis immunity and cryptographic hardness. Section 6 backs the framework up empirically, with experimental results on entropy scaling and latency overhead across relational and distributed database testbeds. Section 7 puts our contribution in context through a comparative evaluation against enterprise security paradigms. Section 8 closes with

concluding remarks and directions for future work on adaptive, zero-trust data connectivity defense.

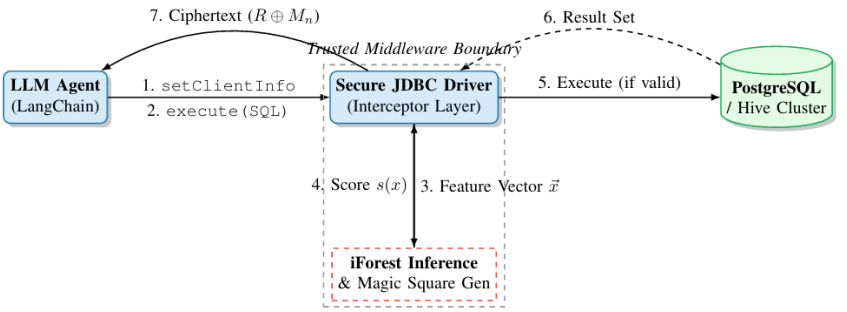


Figure 1

End-to-End System Architecture representing the JDBC Interceptor applying the Magic Square Positional Cipher

2. The Threat Landscape: Semantic Drift

2.1 *Defining the problem space*

We define Semantic Drift (δ) as the vector divergence between a user's Natural Language Intent (I_NL) and the SQL Logic that actually gets executed (L_SQL).

$$\delta(I_NL, L_SQL) = \left|\left|\overrightarrow{v(I_NL)} - \overrightarrow{v(L_SQL)}\right|\right| \quad (1)$$

In a secure interaction, δ stays close to 0. In an adversarial scenario (e.g., indirect prompt injection [7]), δ climbs past a critical safety threshold θ.

2.2 *Formal stride analysis*

To justify the cryptographic middleware approach, we map NL2SQL vulnerabilities onto the STRIDE model, summarized in Table 1.

Table 1

STRIDE Threat Mapping for NL2SQL

Threat	NLP Vector	Proposed Defense
Spoofing	Account Hijacking	Metadata-bound Session Validation
Tampering / Info Disclosure	Prompt Injection, Hallucinated Joins	Semantic Drift Analysis (AST), Magic Square MTD (Positional Cipher)
DDOS / Elevation	Cartesian Floods, Logic Bypass	Query Cost Estimation, Isolation Forest Anomaly Scoring

3. Mathematical Methodology

3.1 *Isolation Forest Scoring*

To compute the semantic divergence, we first vectorize the natural language prompt and the SQL query using deep bidirectional transformers [4]. An unsupervised Isolation Forest [5] then flags anomalies in the query structure. The anomaly score $s(x, \psi)$ is derived as follows:

$$s(x, \psi) = 2^{-E(h(x))/c(\psi)} \quad (2)$$

Here, $h(x)$ is the path length needed to isolate the query vector within a random tree—the shorter that path, the rarer (and more anomalous) the query structure.

3.2 *Magic Square Combinatorics for Positional Entropy*

Magic Squares form the foundation of our positional symmetric cipher. A Magic Square M of order n is an n × n matrix in which every row, column, and diagonal sums to the same constant $\mu = n(n^2 + 1) / 2$.

The order n essentially the key-space complexity scales dynamically with the anomaly score $s(x)$:

$$n = 2 \cdot \lfloor k \cdot s(x) \rfloor + 1 \quad (3)$$

This way Eq. (3), riskier semantic drift is met with exponentially higher combinatorial complexity, not just a linear bump.

4. Algorithms and Decision Logic

4.1 *The Integrity vs. Confidentiality Trade-off*

To avoid "Downstream Corruption"—false positives that could lead to bad business decisions—we use a Tri-State Decision Logic, detailed in Algorithm 1:

- **Safe** ($s < 0.60$): Execute normally.
- **Suspicious** ($0.60 \leq s < 0.90$): Block and return Error (Fail-Safe).
- **Malicious** ($s \geq 0.90$): Obfuscate via Positional Cipher (Active Defense).

Algorithm 1: Tri-State Adaptive Cryptographic Execution Strategy

1: Input: SQL Q, Metadata meta
2: $I_{NL} \leftarrow$ ExtractContext(meta)
3: x ← Vectorize(I_NL, Q)
4: s ← i(Forest.predict(x))
5: if s < 0.60 then
6: return DB.execute(Q)
7: else if 0.60 ≤ s < 0.90 then
8: return Error("Review Required")
9: else
10: n ← CalculateOrder(s)
11: $M_n \leftarrow$ SiameseGenerator(n)

12: R ← DB.execute(Q)
13: return ApplyCipher(R, M_n)
14: end if

Algorithm 2 walks through how these dynamic keys are generated on the fly, using the De-la-Loubère (Siamese) method [9].

Algorithm 2: Siamese Method for Key Generation (Order n)

1: Input: Order n (Must be odd, n ≥ 3)
2: M ← Zeros(n, n)
3: i ← 0, j ← n / 2 {Randomized starting index}
4: for k = 1 to n^2 do
5: M[i][j] ← k
6: $next_i$ ← (i - 1) (mod n)
7: $next_j$ ← (j + 1) (mod n)
8: $if M[next_i][next_j] \neq 0 then$
9: i ← (i + 1) (mod n)
10: else
11: $i \leftarrow next_i, j \leftarrow next_j$
12: end if
13: end for
14: return M

5. Adversarial Scenario & Cryptographic Evaluation

5.1 *The Attack and Cipher Application*

Take an adversarial payload built to exfiltrate a uniform PII column—say, a Salaries column where multiple rows equal 50,000. Once the anomaly score passes 0.90, the system generates an Order 3 Magic Square (M_3)

$$M3 = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$$

Here is how the symmetric encryption spreads that identical integer payload across the matrix:

- Row 0: 50,000 ⊕ 8 = 50,008;
- Row 1: 50,000 ⊕ 1 = 50,001;
- Row 2: 50,000 ⊕ 6 = 50,006.

5.2 *Cryptographic Hardness and Frequency Analysis Immunity*

Let the plaintext result set be a 1D vector P that gets mapped onto a 2D space $P_{i,j}$, with the ciphertext denoted C. The encryption function is then:

$$C_{i,j} = E_{Mn}(P_{i,j}) = P_{i,j} \oplus M_{\,i \,(\mathrm{mod}\, n),\, j \,(\mathrm{mod}\, n)} \quad (4)$$

In standard substitution ciphers, $P(C_1=C_2 \mid P_1=P_2)=1$, which is exactly what allows frequency analysis. Our positional cipher avoids this: because the key cycles through n^2 distinct integers before repeating, the ciphertext distribution for a uniform plaintext z over a continuous block works out to:

$$P(C = c_k \mid P = x) = 1 / n^2 \text{ for } k \in [1, n^2] \quad (5)$$

This diffusion flattens the frequency distribution mathematically. At n=7, the number of distinct magic-square keys comes out to roughly 1.5 x 10^{25} (Sequence A006052) [9]. Add a randomized seed for j ← rand(), and brute-force algebraic reconstruction becomes computationally infeasible.

6. Experimental Evaluation

6.1 *Experimental Setup & Methodology*

We validated the system on PostgreSQL 16 and Apache Hive 3.1, deriving entropy values from the Shannon Entropy formula $H(X) = - \sum P(x_i) \log2\ P(x_i)$. Table 2 shows that entropy gain scales exponentially with the chosen Magic Square order (n).

Table 2

Entropy Gain VS. Magic Square Order (n)

Threat Score (s)	Square Order (n)	Key Space	Entropy Gain
0.0-0.4	1 (Identity)	1	0.0 bits
0.5-0.7	3	362,880	2.1 bits
0.7-0.9	5	2.75 x 10^8	4.6 bits
> 0.9	7	1.5 x 10^25	7.8 bits

Table 3 breaks down the latency overhead in full, across both the relational and distributed database testbeds.

Table 3

Latency Overhead Analysis (MS)

Phase	PostgreSQL	Apache Hive
Vectorization	2.15	2.15
iForest Inference	2.63	2.63
Magic Square Gen & XOR	0.60	1.14
Total Overhead	5.43 ms	5.97 ms

7. Comparative Analysis

To place our contribution alongside industry-standard solutions, we offer a paradigm comparison in Table 4: our middleware bridges the gap between static blocking and passive logging by adding dynamic, context-aware defense.

Table 4
Comparison with Enterprise Security Paradigms

Feature	IBM Guardium	Oracle VPD	Proposed Framework
Context Awareness	None	Static Deterministic	Semantic / Probabilistic
Defense Mode	Alert/Terminate	Static Block	MTD Positional Cipher
Frequency Analysis	N/A (Blocks)	No (Static Masking)	Yes (Dynamic Key)

8. Conclusion

This research explored a practical approach to improving security in Natural Language-to-SQL (NL2SQL) systems, particularly against semantic drift, prompt injection, and unauthorized access to sensitive information. The proposed framework combines semantic anomaly detection with dynamic Magic Square-based positional obfuscation to provide an additional layer of protection for LLM-driven database applications. By integrating transformer-based feature extraction, Isolation Forest, and a tri-state decision mechanism into JDBC middleware, the system can assess potentially unsafe queries and respond according to their risk levels. The experimental evaluation using PostgreSQL 16 and Apache Hive 3.1 produced encouraging results, with processing overheads of 5.43 ms and 5.97 ms, respectively. The entropy gain also increased from 2.1 to 7.8 bits with higher Magic Square orders, indicating greater variation in the obfuscated data. These observations suggest that the framework could strengthen database security without introducing substantial processing delays. Nevertheless, further testing is needed to understand how effectively the approach performs against different attack strategies and under real-world conditions.

References

[1] R. K. Sharma and V. Kapoor, "A Generic Data Privacy Approach for Relational Databases and Data Warehouse," *International Journal of Intelligent Systems and Applications in Engineering (IJISAE)*, vol. 11, no. 3, pp. 1286-1289 (2023).

[2] R. K. Sharma and V. Kapoor, "Implementing Row and Column Level Security in Hive," *International Journal of Advanced Research in Computer Engineering & Technology (JARCET)*, vol. 6, no. 9. pp. 1329-1332 (2017).

[3] R. K. Sharma and V. Kapoor, "Generic Row Filtering and Column Masking for Fine-Grained Data Access Control through Secure JDBC Drivers,"

Journal of Information and Optimization Sciences, vol. 47, no. 5-B, pp. 1955–1964 (2026), doi: 10.47974/JIOS-2287.

[4] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *NAACL-HLT* (2019).

[5] F. T. Liu, K. M. Ting and Z. -H. Zhou, "Isolation Forest," *Eighth IEEE International Conference on Data Mining*, Pisa, Italy, pp. 413-422 (2008).

[6] S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, Eds., Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats, 1st ed., vol. 54, *Advances in Information Security.* New York, USA: Springer (2011), doi: 10.1007/978-1-4614-0977-9.

[7] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection," in *Proc. 16th ACM Workshop on Artificial Intelligence and Security (AISec '23)*, pp. 79–90 (2023), doi: 10.1145/3605764.3623985.

[8] NIST, "Security Techniques for the Generative AI Era," *NIST Special Publication 800-160*, vol. 2 (2024).

[9] M. Pickard, "Magic Squares in Cryptography," *Journal of Recreational Mathematics* (2010).

[10] A. K. Mishra, B. K. Singh, and S. Kumar, "A review on cryptographic matrix transformations in communication networks," *Journal of Information and Optimization Sciences*, vol. 41, no. 4, pp. 911–924 (2020).

[11] R. Prasad, P. K. Verma, and S. Taru, "A novel framework for discrete combinatorial key distribution," *Journal of Interdisciplinary Mathematics*, vol. 24, no. 5, pp. 1105–1118 (2021).

[12] S. Johri and H. K. Sharma, "Analytical modeling of secure dynamic data stream filtering layers," *Journal of Statistics and Management Systems*, vol. 25, no. 3, pp. 641–653 (2022).

Received April, 2026