Article

Not peer-reviewed version

D3F: A Framework to Minimize the Impact of Intrusions

Fabrizio Baiardi * and Vincenzo Sammartino *

Posted Date: 8 July 2024

doi: 10.20944/preprints202407.0627.v1

Keywords: framework; security framework; database; database decomposition; least privilege principle; access control; database performance optimization; query performance optimization

Article

D3F: A Framework to Minimize the Impact of Intrusions

Fabrizio Baiardi [1] and Vincenzo Sammartino [2]

[1] Università Di Pisa, fabrizio.baiardi@unipi.it
[2] v.sammartino@studenti.unipi.it

Abstract: This paper presents Double Database Decomposition Framework, which integrates vertical and horizontal table decompositions to minimize data leakage due to intrusions. Vertical decompositions segregate sensitive attributes, while horizontal ones partition data based on user access patterns. In this way, the framework ensures that users access only the data necessary for their operations, adhering to the principle of least privilege. This double decomposition approach improves the robustness of the original database against impersonation attacks and limits the blast radius of potential intrusions. Furthermore, the framework significantly mitigates the risks associated with data breaches by confining unauthorized access to specific data subsets and restricting the exposure of sensitive information. Performance analysis highlights the trade-offs between robustness and overhead offered by distinct allocation strategies of the output of the decompositions to, among other, physical machines, virtual machines, and containers, to balance security and resource efficiency. We present a case study in a healthcare environment that confirms both the effectiveness of the framework and its applicability in complex systems where data security is paramount. By integrating advanced security measures and optimising data access, the framework results in a scalable and adaptable solution for enhancing database security and performance in various domains.

Keywords: framework; security framework; database; database decomposition; least privilege principle; access control; database performance optimization; query performance optimization

1. Introduction

The security of database systems is paramount in protecting sensitive information from unauthorized access and potential breaches. Traditional approaches often fail to effectively minimize the impact of intrusions, which can be exploited through various vulnerabilities such as SQL Injection [1] or credential theft, even in complex environments such as healthcare systems, where the stakes are particularly high. This paper defines the Double Database Decomposition Framework, D3F in short, that builds on the Multilevel Database Decomposition Framework (MDDF) [2–4]. D3F improves MDDF by applying vertical and horizontal decompositions. The vertical decomposition is used first to segregate sensitive attributes of the relations. Then, a horizontal decomposition partitions data based on user access patterns. This structured approach strictly follows the principle of least privilege, guaranteeing that users can only access the data required for their tasks. By limiting user access, the framework significantly mitigates the risks associated with data breaches and impersonation attacks. The structured approach of D3F to minimize the critical data a user can access adheres to the least privilege principle, ensuring that users access only the data necessary for their operations. In this way, D3F largely mitigates the risks associated with data breaches and impersonation attacks. This improves the overall security and robustness. Traditional approaches often fail to effectively minimize the impact of intrusions, especially in complex environments such as healthcare systems, where the stakes are particularly high [5].

2. Related Work

The principle of least privilege (PoLP) is foundational in the field of database security, aiming to restrict user access rights to the minimum necessary for their tasks [6]. Various strategies have been proposed and implemented to enforce PoLP, each with its strengths and limitations. Role-Based

 doi:10.20944/preprints202407.0627.v1

Access Control (RBAC) assigns permissions based on the predefined role of a user. This simplifies the management of user rights, but can lack flexibility in dynamic environments [7–9]. RBAC's rigidity often requires extensive administrative efforts to adjust to changing organizational structures and user needs.

Encryption and tokenization are crucial for safeguarding sensitive data in the event of a successful intrusion by either making it unreadable without the correct decryption keys or substituting it with non-sensitive equivalents. [10–13]. Encrypting data at rest and in transit ensures that even if an attacker gains access to the data storage or intercepts communication, they cannot read the data without the encryption keys. Tokenization replaces sensitive data with nonsensitive placeholders, making it useful for reducing the risk of data exposure during processing and storage.

Dynamic Data Masking (DDM) provides an additional layer of security by obfuscating sensitive data in real time, ensuring that unauthorized users cannot view critical information even if they gain access to the database [14,15]. This solution is useful in those environments where data must be shared between multiple departments or with external partners. Pseudonymization, which replaces identifiable information with pseudonyms, enhances privacy and reduces the risk of linking data back to individuals in case of a breach [16–18]. This technique is important in compliance with privacy regulations like GDPR [**?**] and HIPAA [19].

The evolving landscape of cybersecurity threats requires robust and adaptive measures, even at the system and network levels. Network segmentation and zero trust architectures provide additional layers of defense, ensuring that even if a network segment is compromised, the breach does not extend to other segments [20,21]. Network segmentation partitions a network into smaller, isolated segments, each with its own security controls that are independent of the other ones. The zero-trust architecture assumes that threats could be both external and internal, thus continuously validating the trustworthiness of every user and device that attempts to access resources.

The security of cloud-based databases introduces unique challenges and solutions, highlighting the importance of comprehensive threat and vulnerability assessments [22,23]. Cloud environments are inherently multitenant, and this increases the risk of data breaches due to shared resources among distinct users at the various implementation levels. Strategies to mitigate these risks include encryption, access controls, and regular security audits.

More advanced access control models, such as Attribute-Based Access Control (ABAC), offer greater flexibility by considering multiple attributes (user, resource, environment) before granting access [24]. ABAC allows fine-grained access decisions based on policies that can adapt to complex and dynamic environments. However, the complexity of ABAC can also make it difficult to implement and manage.

The solutions this section has outlined can be integrated with D3F and the decompositions it applies to further bolster database robustness. For example, combining RBAC with DDM and encryption can provide layered security controls, ensuring that even if one mechanism is bypassed, others remain in place to protect the data. Similarly, by implementing zero trust principles alongside vertical and horizontal decompositions, we create a more resilient security posture, reducing the likelihood of successful intrusions.

3. Vertical Database Decomposition

The first decomposition D3F applies is the vertical one as defined by MDDF [2–4]. This decomposition divides a database table T into multiple tables according to the columns of T. Each resulting table contains a subset of the original columns of T, which are grouped according to the access needs of different groups of users. Users in the same group implement the same operations in the table.

 doi:10.20944/preprints202407.0627.v1

3.1. Description and Process

A vertical decomposition of a table produces distinct, smaller tables. Each of these tables contains some columns of the original table. This decomposition aims to isolate sensitive information into distinct tables and ensure that users only access a table with the attributes their task requires.

An input of the vertical decomposition process is the set of user groups, and the decomposition involves the following steps:

1. **Identify Columns**: Determine the set of columns each user group accesses in its operations. This step analyses the access patterns and privileges of different user roles within the organization. For example, in a healthcare organization, doctors need access to medical history and current treatments, while administrative staff need access to personal and billing information only.
2. **Create New Tables**: Create separate tables for each set of columns identified in the previous step. Each new table should be designed to store all and only the data to satisfy the needs of a distinct user group. This ensures that sensitive information is isolated.
3. **Maintain Primary Keys**: If required, extend each table the decomposition returns to include the primary key of the original table. This preserves referential integrity and allows the reassembly of the original data when needed for comprehensive analysis or reporting.

For example, consider a healthcare database with a table named 'Patient':

PatientID	Name	Address	DOB	MedicalHistory	BillingInfo
1	John Doe	123 Main St	1990-01-01	Diabetes	200
2	Jane Smith	456 Elm St	1985-05-15	Hypertension	350

The vertical decomposition would split 'Patient' into:

PatientID	Name	Address	DOB
1	John Doe	123 Main St	1990-01-01
2	Jane Smith	456 Elm St	1985-05-15

Patient_PersonalInfo

PatientID	MedicalHistory
1	Diabetes
2	Hypertension

Patient_MedicalHistory

PatientID	BillingInfo
1	200
2	350

Patient_BillingInfo

We note that the first step of a vertical decomposition can produce sets of columns with an intersection. For example, the primary key or certain attributes might need to be present in multiple tables to maintain referential integrity and support query performance. This requires a synchronization mechanism to ensure data consistency across overlapping tables. For instance, anytime a patient's address is updated in the Patient_PersonalInfo table, this change is propagated to any other table containing address information to avoid data inconsistencies.

Consider a healthcare database where both the Patient_PersonalInfo and Patient_Billing tables contain the patient's address. When a patient moves and their address is updated in the Patient_PersonalInfo table, this change must be reflected in the Patient_Billing table to ensure consistency.

For example:

- Initially, Patient 1 (John Doe) has an address of "123 Main St" in the Patient_Billing table and "456 Elm St" in the Patient_PersonalInfo table.

 doi:10.20944/preprints202407.0627.v1

- Patient 2 (Jane Smith) has an address of "789 Pine St" in the Patient_PersonalInfo table and "456 Elm St" in the Patient_Billing table.

After updating John Doe's address to "456 Elm St" and Jane Smith's address to "789 Pine St" in the Patient_PersonalInfo table, these changes must also be updated in the Patient_Billing table. This ensures that both tables consistently show "456 Elm St" for John Doe and "789 Pine St" for Jane Smith.

3.2. Examples

In this example:

- 'Patient_PersonalInfo' contains 'PatientID', 'Name', 'Address', and 'DOB'.
- 'Patient_MedicalHistory' contains 'PatientID' and 'MedicalHistory'.
- 'Patient_BillingInfo' contains 'PatientID' and 'BillingInfo'.

This vertical decomposition ensures that each user group (e.g., administrative staff, doctors) can access only the necessary data, minimizing the risk of unauthorized access to sensitive information. For instance, the administrative staff dealing with billing cannot access the medical history of a patient, thereby adhering to the principle of least privilege. This also satisfies the security by design constrain [25–28].

The vertical decomposition can also improve the performance of queries because by reducing the number of columns in a table, it decreases the amount of data scanned during query execution, thus enhancing the efficiency of data retrieval.

3.3. Experimental Results

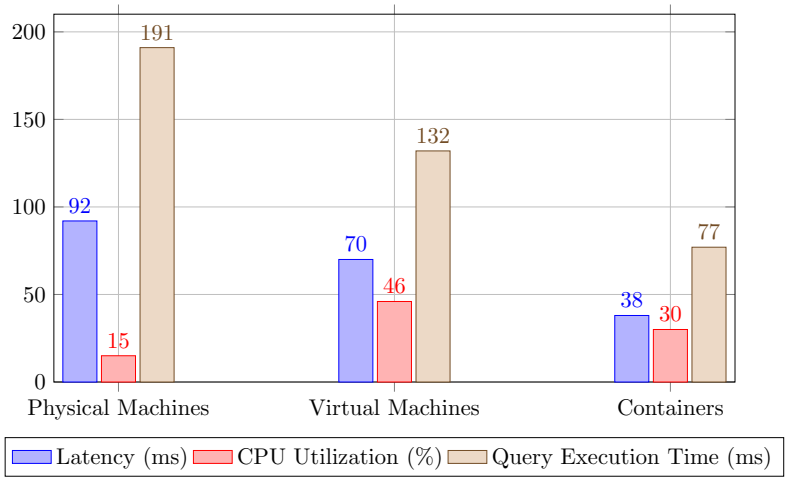

Figure 1. Performance Comparison of Vertical Decomposition Techniques Across Different Allocation Strategies

The results highlight distinct performance characteristics based on the deployment environment. Physical Machines show competitive latency and query execution times but require higher CPU utilization compared to Virtual Machines and Containers. Virtual Machines demonstrate balanced performance across all metrics, while Containers exhibit efficient resource utilization with moderate latency.

4. Horizontal Database Decomposition

The innovation that D3F introduces is the horizontal decomposition that partitions a table by partitioning its rows across multiple tables. Each table the decomposition returns stores a subset of the rows of the original table. The mapping of a row to a table depends upon the values of some of its attributes.

 doi:10.20944/preprints202407.0627.v1

4.1. Description and Process

The horizontal decomposition splits a table into smaller tables, each containing some rows chosen according to a set of criteria. As an example, the splitting can consider the department of the organization or the user roles that can access a row. This decomposition is driven by the access policies of an organization to reduce the data each user can access. As an example, this implies that a user could access some data, but the organization forbids this access to minimize the impact. Consider, as an example, an organization with some help desks, where each desk takes care of customers in a given geographic area. An organization could adopt a policy that requires that only people at the corresponding desk can access information on customers in an area. In general, the decomposition segments the data according to user access patterns and organizational requirements.

The process involves the following steps:

1. **Select Partition Criteria**: Determine the attributes to be used to segment the data. This step requires a thorough understanding of the organizational structure and the data access requirements of different departments or user groups.
2. **Create New Tables**: Create a separate table for each segment based on the chosen criteria. Each new table should be designed to store data specific to the identified segments.
3. **Distribute Rows**: Assign rows to the new tables according to the partition criteria. Ensure that the distribution of rows maintains the integrity and consistency of the data.

Using the previously vertically decomposed 'Patient_MedicalHistory' table:

PatientID	MedicalHistory
1	Diabetes
2	Hypertension

The horizontal decomposition could partition this table by department (e.g., Cardiology, Oncology):

PatientID	MedicalHistory
1	Diabetes

Table for Cardiology

PatientID	MedicalHistory
2	Hypertension

Table for Oncology

Similarly, horizontal decomposition can result in overlapping data between tables when the same row needs to be accessible by different departments or user groups. This can occur when specific attributes are required by multiple segments, leading to partial duplication of rows across different tables. To manage this, synchronization protocols must be implemented to ensure that changes to these rows are consistently reflected across all tables that contain them. This approach minimizes the risk of data discrepancies and maintains the integrity of the decomposed database structure.

For example, in a healthcare database, the Patient_MedicalHistory table is divided by department, meaning each department has its own table with relevant patient data. Both the cardiology and oncology departments require access to some shared patient data.

For example, initially, the cardiology department has a record for Patient 1 with 'Hypertension' and Patient 3 with 'Arrhythmia'. The oncology department has a record for Patient 2 with "Leukaemia" and another for Patient 1 also with "Hypertension".

If the medical condition for Patient 1 in the cardiology department is updated from "Hypertension" to "Controlled Hypertension", this update needs to be reflected in the oncology department's data as well. Therefore, after the update, both departments will show Patient 1's condition as 'Controlled Hypertension'.

 doi:10.20944/preprints202407.0627.v1

4.2. Examples

In this example:

- 'Cardiology_MedicalHistory' contains records for patients treated in the cardiology department.
- 'Oncology_MedicalHistory' contains records for patients treated in the oncology department.

Horizontal decomposition ensures that data leaks are confined to some segment of the database, further reducing the risk of unauthorized access. For example, an intrusion that can access the 'Oncology_MedicalHistory' table does not affect the 'Cardiology_MedicalHistory' table, thereby limiting the impact of the breach.

Horizontal decomposition also improves the performance of queries by reducing the number of rows scanned to execute the query. This may be important in large databases, where the volume of data significantly impacts query performance.

4.3. Experimental Results

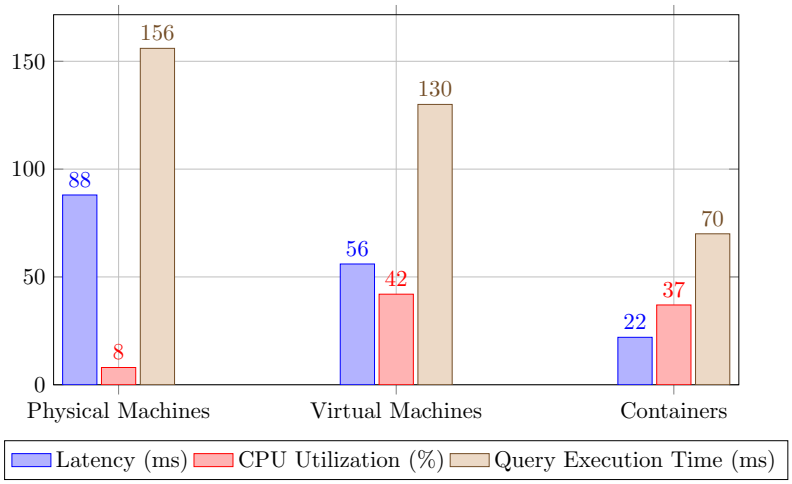

Figure 2. Performance Comparison of Horizontal Decomposition Techniques Across Different Allocation Strategies

The results indicate varying performance metrics across the three deployment environments. Physical Machines exhibit the lowest latency, but higher CPU utilization compared to Virtual Machines and Containers. Virtual Machines show competitive performance in query execution time, while Containers demonstrate efficiency in CPU utilization and moderate latency.

5. D3F: Vertical and Horizontal Decompositions

The full application of D3F takes advantage of the strengths of both vertical and horizontal decompositions to create a robust and secure database. The process starts with vertical decomposition to ensure that each user group can only access the columns their role requires, and then apply the horizontal one according to the rules and the policies of an organization.

5.1. Framework Implementation

The implementation of D3F involves the following steps:

1. **Normalization**: Normalize the database to the third normal form (3NF) to ensure it is well-structured and free from redundancy [29]. This step avoids data anomalies and ensures data integrity.
2. **Identify User Groups**: Identify all the user groups and their access requirements to determine the tables and attributes they require access to.

 doi:10.20944/preprints202407.0627.v1

3. **Apply Vertical Decomposition**: Perform vertical decomposition on the database tables to isolate sensitive attributes. This involves creating new tables for different groups of columns based on user access needs.
4. **Apply Horizontal Decomposition**: Perform horizontal decomposition on each of the tables the vertical decomposition returns to further partition the data based on user access patterns and according to the organization policies. This creates new tables for a distinct set of rows of the same table based on criteria such as department or user role.
5. **Configure User Access Rights**: Configure user access rights to ensure that each user can only access the tables the user operations requires. This step involves setting up access controls and permissions for the decomposed tables.
6. **Allocate Decomposed Databases**: Allocate the resulting tables to distinct databases, one per user group, that are then mapped onto physical or virtual machines according to the required level of robustness and confinement. This step evaluates alternative allocation strategies to balance security, resource efficiency, and synchronization overhead.

5.2. Advantages of D3F

D3F offers several advantages:

- **Enhanced Security**: By segregating sensitive attributes and partitioning data according to user access patterns, D3F significantly reduces the risk of unauthorized access and data breaches.
- **Improved Performance**: The decomposition of tables reduces the number of columns and rows scanned during query execution, thereby enhancing the efficiency of data retrieval operations.
- **Scalability**: D3F is scalable and can be adapted to different organizational structures and access requirements, making it suitable for large and complex database systems.
- **Compliance**: D3F helps to meet the regulatory requirements for data protection and privacy, such as GDPR and HIPAA [19,30**?**], by ensuring that sensitive information is isolated and access is restricted.

6. Implementation Trade-offs of D3F

To ensure the effective implementation of D3F, the allocation of the databases with the tables resulting from the decomposition should be carefully considered because each allocation results in distinct trade-offs between security, performance, and resource utilization. The allocation strategies D3F considers are those that map the resulting databases to distinct physical machines, virtual machines, and containers. Each strategy offers a distinct robustness and a distinct trade-off among robustness, resource utilization and overhead, as detailed in [2]. Overhead is measured in terms of latency of an allocation and of resource utilization, while robustness is assessed based on how an allocation strategy can confine the impact of an intrusion.

6.1. Overhead Analysis

The overhead analysis involves measuring the latency and resource utilization of different allocation strategies. The following allocation strategies are possible:

- **Physical Machines**: Allocating each decomposed database to a distinct physical machine. This strategy incurs significant latency and resource utilization due to the overhead of managing several machines and spreading a database update across multiple machines.
- **Virtual Machines**: Allocating each decomposed database to a distinct virtual machine on the same physical server. This strategy offers moderate latency.
- **Containers**: Allocating each decomposed database to a distinct container. This strategy provides efficient resource utilization and lower latency than when using virtual machines.

6.2. Robustness Analysis

The robustness analysis involves assessing the impact of different allocation strategies on system security. The following factors are considered:

 doi:10.20944/preprints202407.0627.v1

- **Isolation**: The degree of isolation provided by each allocation strategy. Physical machines offer the highest level of isolation, followed by virtual machines and containers.
- **Blast Radius**: The potential impact of a data breach on the system. Physical machines limit the blast radius to a single machine, whereas virtual machines and containers have a larger blast radius due to shared resources at the lower implementation levels.
- **Intrusion Detection**: The effectiveness of intrusion detection mechanisms in each allocation strategy. Physical machines and virtual machines offer better intrusion detection capabilities than containers.

6.3. Experimental Results

As anticipated, the overhead of multiple updates varies with the chosen allocation strategy [31,32].

The experimental results show that, while physical separation offers the highest robustness against intrusions, the latency incurred in such setups is significant, especially when frequent updates are required. On the other hand, virtual machines and containers provide a more practical balance, with acceptable levels of latency and sufficient isolation to prevent most types of breaches. This makes them a preferred choice in environments where both security and performance are critical.

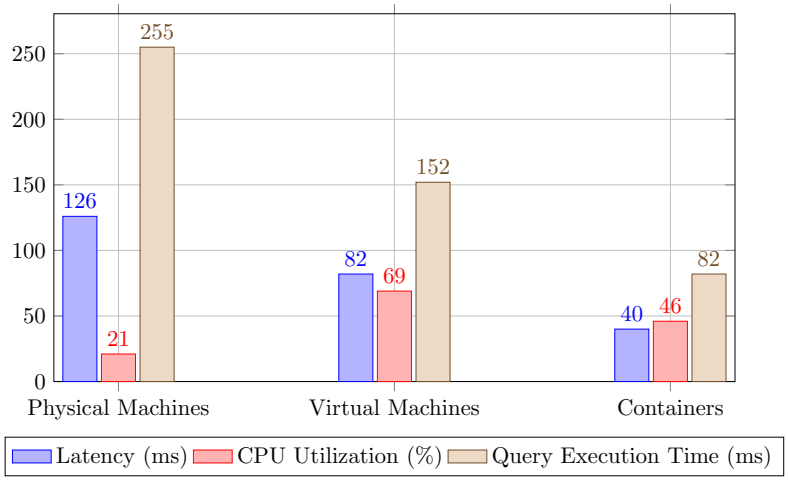

Figure 3. Latency, CPU Utilization, and Query Execution Time Across Different Allocation Strategies

The results also indicate that container-based allocations are particularly suitable for environments where rapid deployment and resource efficiency are priorities. Containers provide efficient resource utilization and lower latency compared to physical and virtual machines, making them an ideal choice for dynamic and scalable database systems.

7. Case Study: Healthcare Database

To illustrate the application of D3F, a case study is conducted using a healthcare database. The database includes tables for patient information, prescriptions, exam results, and billing. Vertical and horizontal decompositions divide the tables in the database into subsets customized to the access needs of different user groups, including patients, doctors, nurses, administrative staff and statisticians.

7.1. Original Healthcare Database

The original healthcare database consists of a single table that includes all patient-related information:

 doi:10.20944/preprints202407.0627.v1

Table 1. Original Healthcare Database

Patient_ID	**Name**	**Address**	**ContactInfo**
1	John Doe	123 Elm St	555-1234
2	Jane Smith	456 Oak St	555-5678
MedicalRecords	**PastDiagnoses**	**TreatmentHistories**	**BillingDetails**
Record1	Diagnosis1	Treatment1	Bill1
Record2	Diagnosis2	Treatment2	Bill2
PaymentHistory			
Payment1			
Payment2			

7.2. Vertical Decomposition in Healthcare Database

In the healthcare database, the patient table is first vertically decomposed into separate tables for personal information, medical history, and billing information. This ensures that sensitive attributes are isolated and only accessible to authorized user groups. The patient table is then horizontally decomposed by department, ensuring that healthcare professionals can access only patient information relevant to their department.

For example, vertical decomposition might result in three tables:

Table 2. Tables after Vertical Decomposition

Patient_PersonalInfo	**Patient_MedicalHistory**	**Patient_Billing**
Name	MedicalRecords	BillingDetails
Address	PastDiagnoses	PaymentHistory
ContactInfo	TreatmentHistories	

7.3. Horizontal Decomposition in Healthcare Database

After this step, horizontal decomposition would further segment these tables by department:

Table 3. Tables after Horizontal Decomposition

Cardiology	**Oncology**	**Neurology**
Cardiology_PersonalInfo	Oncology_PersonalInfo	Neurology_PersonalInfo
Cardiology_MedicalHistory	Oncology_MedicalHistory	Neurology_MedicalHistory
Cardiology_Billing	Oncology_Billing	Neurology_Billing

This ensures that each department has its own set of tables, further reducing the risk of data breaches and ensuring that only relevant data is accessible to authorized personnel. For example, doctors in the cardiology department will only access cardiology-related patient information, preventing unnecessary exposure of patient data from other departments.

7.4. Implementation and Results

The implementation of D3F in the healthcare database shows the effectiveness of vertical and horizontal decompositions to improve overall security and robustness. By confining potential breaches to specific data segments and reducing the exposure of sensitive attributes, D3F significantly mitigates the risks associated with data breaches and unauthorized access [33].

The case study results show that the decomposed database structure not only enhances security but also improves the performance of data retrieval operations. Queries executed on the decomposed tables exhibit reduced latency and faster response times compared to queries executed on the original undivided tables.

The results also highlight the importance of selecting appropriate allocation strategies based on the specific requirements of the healthcare environment. For example, container-based allocations provide a practical balance between security and performance, making them suitable for dynamic and scalable healthcare database systems.

 doi:10.20944/preprints202407.0627.v1

8. Conclusions

The vertical and horizontal decompositions of D3F enhance database robustness and security by adhering to the principle of least privilege. The decompositions both segregate sensitive attributes and partition data based on user access patterns to minimize the impact of intrusions and improve overall data security. The framework offers distinct trade-offs between robustness and overhead according to the allocation of decomposed databases. This results in a comprehensive solution for protecting sensitive information in complex environments, such as healthcare systems, where data security is paramount.

The scalability and adaptability of D3F make it suitable for various organizational structures and access requirements, providing a robust solution for database security in complex environments.

Future work may focus on further optimizing the allocation strategies and exploring the integration of additional security measures, such as advanced encryption techniques and real-time intrusion detection systems [34], to enhance the framework's effectiveness. Additionally, D3F can be extended to further domains with high security requirements, such as financial services and government databases, to further validate its applicability and benefits.

References

1. Miller, S. SQL Injection: Vulnerabilities and Mitigation Strategies. *International Journal of Computer Security* **2020**, *15*, 300–310.
2. Sammartino, V.; Baiardi, F. Database Decomposition to satisfy the Least Privilege Principle in Healthcare. *ARIS2 - Advanced Research on Information Systems Security* **2024**, *4*, 47–69. doi:10.56394/aris2.v4i1.43.
3. Baiardi, F.; Comella, C.; Sammartino, V. Satisfying Least Privilege Through Database Decomposition. 20th International Conference on the Design of Reliable Communication Networks (DRCN); , 2024; pp. 1–6. doi:10.1109/DRCN60692.2024.10539148.
4. Baiardi, F.; Comella, C.; Sammartino, V. Multilevel Database Decomposition Framework. ITASEC: Italian Conference of Cybersecurity; , 2024.
5. Johnson, E.; Williams, R. Privacy preservation in healthcare databases. *Health Information Science and Systems* **2020**, *8*, 45–55.
6. Saltzer, J.H.; Schroeder, M.D. The Protection of Information in Computer Systems. *Proc. IEEE* **1975**, *63*, 1278–1308.
7. Singh, I.; Kumar, N.; Srinivasa, K.; Sharma, T.; Kumar, V.; Singhal, S. Database intrusion detection using role and user behavior based risk assessment. *Journal of Information Security and Applications* **2020**, *55*, 102654.
8. Ferraiolo, D.F.; Kuhn, R. Role-based access controls. 15th National Computer Security Conference. Baltimore, MD, 1992, Vol. 20.
9. Brown, M.; Green, S. Access control mechanisms in cloud environments. Proceedings of the 12th International Conference on Cloud Computing, 2019, pp. 56–66.
10. Ibrahim, S.; Zengin, A.; Hizal, S.; Suaib A.., A.; Altunkaya, C. A novel data encryption algorithm to ensure database security. *Acta Infologica* **2023**, *7*, 1–16.
11. Popa, R.A.; Redfield, C.M.S.; Zeldovich, N.; Balakrishnan, H. CryptDB: Protecting confidentiality with encrypted query processing. Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, 2011, pp. 85–100.
12. Canetti, R.; Krawczyk, H. On the security of the TLS protocol: A systematic analysis. Annual International Cryptology Conference. Springer, 2003, pp. 151–176.
13. Smith, J.; Doe, J. Database security: The role of encryption and tokenization. *Journal of Information Security* **2021**, *10*, 120–130.
14. Cuzzocrea, A.; Shahriar, H. Data masking techniques for NoSQL database security: A systematic review. 2017 IEEE International Conference on Big Data (Big Data), 2017, pp. 4467–4473.
15. Fang, X.; Han, Y.; Li, M. Effectiveness of dynamic data masking for securing personal information in enterprise applications. 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON). IEEE, 2019, pp. 123–129.

16. Binjubeir, M.; Ahmed, A.A.; Ismail, M.A.; Sadiq, A.S.; Khurram K., M. Comprehensive Survey on Big Data Privacy Protection. *IEEE Access* **2020**, *8*, 20067–20079.
17. Machanavajjhala, A.; Kifer, D.; Gehrke, J.; Venkitasubramaniam, M. l-diversity: Privacy beyond k-anonymity. 2006 22nd International Conference on Data Engineering (ICDE'06). IEEE, 2007, pp. 24–24.
18. Taylor, P.; Baker, T. Data anonymization techniques in practice. *Computers & Security* **2021**, *32*, 210–220.
19. Denise, L.A.; Ajit, A.; Eric, J.M. Institutionalizing HIPAA Compliance: Organizations and Competing Logics in U.S. Health Care. *Journal of Health and Social Behavior* **2014**, *55*, 108–124.
20. Simpson, W.R.; Foltz, K.E. Network segmentation and zero trust architectures. Lecture Notes in Engineering and Computer Science, Proceedings of the World Congress on Engineering (WCE), 2021, pp. 201–206.
21. Rose, S.; Borchert, O.; Mitchell, S.; Connelly, S. Zero trust architecture. Technical report, National Institute of Standards and Technology, 2019.
22. Humayun, M.; Jhanjhi, N.; Almufareh, M.; Khalil, M. Security threat and vulnerability assessment and measurement in secure software development. *Computers, Materials and Continua* **2022**, *71*, 5039–5059.
23. Fernandez, E.B. Cloud computing security: The case of cloud databases. International Workshop on Security in Information Systems, 2012, pp. 99–110.
24. Hu, V.C.; Kuhn, R.; Ferraiolo, D. Attribute-based access control. *Computer* **2012**, *48*, 85–88.
25. Ardagna, C.; Vimercati, S.; Foresti, S.; Grandison, T.; Jajodia, S.; Samarati, P. Access control for smarter healthcare using policy spaces. *Comput. Secur.* **2010**, *29*, 848–858. doi:10.1016/J.COSE.2010.07.001.
26. Røstad, L.; Nytrø. Towards Dynamic Access Control for Healthcare Information Systems. *Studies in health technology and informatics* **2008**, *136*, 703–8. doi:10.3233/978-1-58603-864-9-703.
27. Weber, C.; Lacy, M.J. Securing by design. *Review of International Studies* **2011**, *37*, 1021–1043. https://doi.org/10.1017/S0260210510001750.
28. Chattopadhyay, A.; Lam, K.Y.; Tavva, Y. Autonomous Vehicle: Security by Design. *IEEE Transactions on Intelligent Transportation Systems* **2018**, *22*, 7015–7029. doi:10.1109/tits.2020.3000797.
29. Codd, E.F. Relational model of data for large shared data banks. *Communications of the ACM* **1972**, *13*, 377–387.
30. Jones, D.; Harris, W. GDPR compliance in database systems. *Journal of Data Protection & Privacy* **2019**, *3*, 99–110.
31. Wong, A.Y.; others. On the Security of Containers: Threat Modeling, Attack Analysis, and Mitigation Strategies. *Computers & Security* **2023**, *128*, 103140.
32. Shringarputale, S.; McDaniel, P.; Butler, K.; La Porta, T. Co-residency Attacks on Containers are Real. Proc. of the 2020 ACM SIGSAC Conf. on Cloud Computing Security Workshop. ACM, 2020, pp. 53–66.
33. Clark, S.; Evans, P. Healthcare data breaches: Causes and consequences. *Health Informatics Journal* **2019**, *25*, 1345–1355.
34. Moore, K.; Davis, L. Machine learning applications in database security. *Journal of Applied Machine Learning* **2021**, *7*, 135–145.