Balancing Privacy and Efficiency: Music Information Retrieval via Additive Homomorphic Encryption

William Zerong Wang
Independent Researcher
s-wangwil@outlook.com

Dongfang Zhao
University of Washington
dzhao@cs.washington.edu

Abstract

In the era of generative AI, ensuring the privacy of music data presents unique challenges: unlike static artworks such as images, music data is inherently temporal and multimodal, and it is sampled, transformed, and remixed at an unprecedented scale. These characteristics make its core vector embeddings, i.e, the numerical representations of the music, highly susceptible to being learned, misused, or even stolen by models without accessing the original audio files. Traditional methods like copyright licensing and digital watermarking offer limited protection for these abstract mathematical representations, thus necessitating a stronger, e.g., cryptographic, approach to safeguarding the embeddings themselves. This line of works aims to develop a cryptographic framework for music embeddings that balances security and efficiency, as supporting core functionalities like music recommendation and retrieval requires the ability to perform expensive similarity searches on encrypted data. Standard encryption schemes, such as AES, render data unintelligible for computation, making such searches impossible. While Fully Homomorphic Encryption (FHE) provides a plausible solution by allowing arbitrary computations on ciphertexts, its substantial performance overhead remains impractical for large-scale vector similarity searches. Given this trade-off, we propose a more practical approach using Additive Homomorphic Encryption (AHE) for vector similarity search. Although AHE supports a restricted set of operations, i.e., only arithmetic additions among ciphertexts, we show that under certain threat models those additive operations suffice to construct scalar multiplications and provide provable security in common scenarios like safeguarding user query privacy or database confidentiality, all at a significantly lower computational cost than FHE. The primary contributions of this paper are threefold: we analyze threat models unique to music information retrieval systems; we provide a theoretical analysis and propose an efficient AHE-based solution through

inner products of music embeddings to deliver privacy-preserving similarity search; and finally, we demonstrate the efficiency and practicality of the proposed approach through empirical evaluation and comparison to FHE schemes on real-world MP3 files.

1 Introduction

In the era of generative AI, music data presents increasingly complex privacy challenges. Recent advancements such as OpenAI's Jukebox [1] and Google's MusicLM [2] demonstrate that high-fidelity musical content can now be generated, conditioned on genre, instrumentation, or even textual prompts. These systems rely on learned vector embeddings to power tasks such as synthesis, classification, and recommendation. However, the same embeddings also introduce vulnerabilities: once exposed, they can be sampled, remixed, or reverse-engineered by generative models without requiring access to the original audio. As the use of music embeddings expands across information retrieval pipelines, recommendation systems, and creative tools, it becomes increasingly important to ensure that these representations are protected from misuse or unauthorized replication [3, 4, 5]. Balancing accessibility with protection is therefore a central challenge for secure and ethical machine learning on music data.

Recent high-profile incidents highlight the urgency of this issue. In April 2023, the viral AI-generated song "Heart on My Sleeve" mimicked the voices of Drake and The Weeknd using unauthorized voice cloning and generative AI techniques. The track garnered millions of views before being removed at the request of Universal Music Group, which condemned it as a copyright violation [6]. This case illustrates how AI can exploit original music data to create realistic but infringing material. Although from a different medium, the Studio Ghibli AI art controversy provides a valuable parallel [7]. Furthermore, in the context of remixes, mashups, and other creative reinterpretations, it is essential to preserve the integrity of the original musical components. While derivative works are a vital and culturally rich part of music evolution, the original content, especially when embedded as vectors, must be safeguarded from unauthorized extraction, misuse, or misattribution. This is especially important for creators who share portions of their work collaboratively but seek to retain control over core melodies, harmonies, or expressive performance elements.

Equally important is the protection of user music preferences that can be derived from queries during music information retrieval (IR). Music taste can reflect deeply personal or emotional states, and listening data is often used

for therapy, mood regulation, or identity exploration. Many users would not want such preferences to be exposed or exploited. Ensuring the privacy of music queries upholds user dignity and autonomy.

Traditional methods like copyright licensing and digital watermarking offer limited protection for these abstract mathematical representations, thus necessitating a cryptographic approach to directly safeguard the embeddings themselves. Vector similarity search is a fundamental operation in modern machine learning and music IR applications, such as Shazam—a music search application [8], and Spotify, which uses vector databases for recommendation [9]. These vector databases [10] are also widely used in recommendation systems, retrieval-augmented generation, and large-scale search [11]. To ensure data privacy in these settings, Fully Homomorphic Encryption (FHE) has been extensively studied for encrypted similarity search; however, it remains computationally expensive and impractical for real-time or large-scale applications [12, 13].

We observe that in many scenarios, applying FHE is an overkill because the multiplication between two ciphertexts is unnecessary. For example, recent works [14, 15] demonstrated that only one component—either the stored database vectors or the query vector—requires encryption, but not both. Music IR systems share the same pattern, where only the query or the music database may need to remain encrypted depending on the specific privacy requirements of the application.

In this paper, we explore the feasibility of performing music vector similarity search using only Additively Homomorphic Encryption (AHE), which supports addition and scalar multiplication in the encrypted domain. Unlike FHE, which enables arbitrary computations at a high cost, AHE provides a significantly more efficient alternative by allowing direct computation of inner products without expensive ciphertext-ciphertext multiplications or bootstrapping. We present three main contributions in: (1) analyzing the threat models of music information retrieval and the reasons behind these threat models; (2) proposing an AHE-based solution for music data retrieval based on the threat models; and (3) demonstrating the practicality of AHE through empirical evaluation.

2 Related work

Vector databases have gained significant attention with the rise of applications such as natural language processing, recommendation systems, and information retrieval, where large-scale vector similarity searches are funda-

mental. Various systems have been designed to support these needs. For instance, Pinecone [16] provides efficient vector search capabilities but often lack flexibility in handling non-vector queries or supporting mixed workloads. These systems primarily rely on approximate nearest neighbor (ANN) search techniques, with graph-based methods such as Hierarchical Navigable Small Worlds (HNSW) [17] being widely adopted [18].

Among purpose-built vector database management systems, Milvus [11] distinguishes itself by supporting multiple index types and hybrid queries, allowing structured attributes to be combined with vector-based similarity searches. This flexibility enables Milvus to integrate traditional query optimization techniques with emerging data modalities, including unstructured text and images. Other systems, such as SingleStore-V [19], further enhance vector search capabilities by incorporating features such as predicate-based vector search and cost-based optimization, making them suitable for federated and distributed environments.

Despite these advancements, many existing solutions lack efficient mechanisms for preserving privacy in distributed and federated learning systems. This limitation is particularly critical for real-time applications, where cryptographic protocols often introduce significant computational overhead [15]. Although prior work on privacy-preserving machine learning establishes foundational approaches, our proposal aims to enhance vector data management by integrating secure and efficient cryptographic mechanisms in music IR systems.

For secure computation, FHE has been widely explored, but has high computational costs that limit its applicability in large-scale retrieval systems. Recent work has demonstrated the feasibility of using AHE for similarity search, significantly reducing computational overhead [20]. Currently, to our knowledge, no existing work applies AHE-based encrypted similarity search to music vector databases, which often contain multimodal and temporal structures. This work aims to design an efficient privacy-preserving retrieval system specifically for music embeddings.

3 Unique Technical Challenges in Music Information Retrieval

First, music data require higher-dimensional embeddings to effectively represent the added complexity. Music is temporal: its sequential structure matters, and thus longer vectors are typically needed to capture this information [21, 22]. Music often involves hierarchy and layering. For example,

orchestral music comprises multiple instruments playing simultaneously, while remix culture overlays tracks from multiple sources. The additional characteristics of music further complicate its retrieval. Pitch invariance, where melodies are recognized regardless of their absolute key, demands transposition-robust embeddings [23]. Rhythm sensitivity must be preserved to distinguish between stylistic and tempo-based variations [21]. Audio quality differences, resulting from compression or recording equipment, also distort feature extraction, requiring embeddings to be resilient to such noise [24]. Cultural variation in musical structure and tuning across traditions add further representational burden. Lastly, semantic fuzziness (e.g, genre or mood) is difficult to represent precisely, necessitating embeddings that generalize from subjective tags or crowd-labeled data [22].

Second, streaming and interactive music IR applications demand extremely low response latency. For example, it typically only takes Shazam around 5 seconds to identify a song, despite having millions of entries in its database [25]. These systems must remain responsive under high load and rapid adoption, underscoring the need for scalable vector search [24].

Both the high-dimensional representation of music data and the requirement for low response latency in large-scale IR systems create unique technical challenges when introducing privacy-preserving mechanisms into music IR. Encryption and decryption of vectors are typically computationally expensive. Thus, it is essential to identify methods that are both effective in preserving privacy and efficient enough for real-world deployment.

4 Homomorphically-Encrypted Music-Embedding Similarity Search

4.1 Threat Model for Music Embeddings

4.1.1 Melody and Rhythm Pattern Inference

In the context of music IR, the threat of privacy breaches extends beyond the simple recovery of an entire encrypted vector. A more subtle and realistic threat is the inference of specific, high-value musical patterns, such as a copyrighted melody, a unique rhythmic signature, or a distinctive chord progression, even when they are part of a larger, fully encrypted work. This threat model assumes an adversary who does not necessarily need the full song, but rather seeks to confirm the presence of a specific, exploitable musical element within an encrypted database. This scenario is highly relevant in an era where AI models can be trained on such patterns to generate new, often

infringing, content, as seen in real-world controversies involving AI-generated music.

The attack scenario can be described as follows, adhering to the formal threat model where the adversary knows the plaintext query $\mathbf{x}$, the encrypted database vector $Enc(\mathbf{y})$, and observes the encrypted result $Enc(s)$:

1. **Target Pattern Identification:** The adversary first identifies a target musical pattern of interest. This could be, for example, the iconic four-note opening of a famous symphony or the chorus melody of a chart-topping pop song.

2. **Query Vector Formulation:** The adversary crafts a plaintext query vector $\mathbf{x}$ that specifically represents this target pattern. For dimensions of the vector that do not correspond to the pattern, the values can be set to zero or neutral noise, effectively isolating the pattern of interest. This leverages the understanding that music embeddings often capture sequential or hierarchical information, making such targeted queries possible.

3. **Similarity Computation:** The adversary performs a similarity search using their crafted query $\mathbf{x}$ against an encrypted music vector $\mathbf{y}'$ from the database. This computation, as described in this paper, results in an encrypted similarity score $s' = Enc(\mathbf{x} \cdot \mathbf{y}')$.

4. **Inference from the Result:** The crucial information leak occurs at this stage. If the adversary is the service provider who also holds the decryption key (a common "honest-but-curious" threat model), they can decrypt the score to get $s = \mathbf{x} \cdot \mathbf{y}'$. A high value of s strongly implies that the encrypted track $\mathbf{y}'$ contains the musical pattern represented by $\mathbf{x}$. Conversely, a low score implies its absence. This allows the adversary to effectively "scan" an entire encrypted library for a specific musical component without ever decrypting the full tracks themselves. Even without the decryption key, the ability to observe and compare the distribution of different $Enc(s)$ values might allow statistical inference over time.

This threat is significant because it directly enables the kind of data exploitation that current copyright and watermarking techniques struggle to prevent for abstract vector embeddings. It allows a malicious actor to systematically identify and harvest core musical ideas for unauthorized use in generative models, fundamentally undermining the privacy and ownership of creative musical works.

4.1.2 Creator Identity Inference

Beyond inferring musical content, a sophisticated adversary may seek to infer the identity of a creator from an encrypted database. This threat model addresses the critical challenge of provenance and attribution in the age of AI-generated content and complex collaborations. The adversary's goal is not to decrypt a song, but to link a piece of music of disputed origin to a specific creator whose works are stored in the database, thereby performing an identity inference attack. This is particularly relevant in copyright disputes or cases of suspected plagiarism involving generative AI.

The attack scenario unfolds as follows:

1. **Disputed Material as Query:** The adversary begins with a piece of music whose origin is in question, for instance, a viral AI-generated track that mimics a famous artist's style, or a melody from a new song that sounds suspiciously familiar. The adversary computes the vector embedding of this disputed track, which becomes their plaintext query vector, $\mathbf{x}$.

2. **Targeted Database Search:** The adversary gains access to perform similarity searches against an encrypted database. This database is assumed to contain the vector embeddings of original works from a known set of creators, where each encrypted vector $Enc(\mathbf{y})$ is implicitly linked to its creator (e.g., stored in a folder designated for Artist A, Artist B, etc.).

3. **Iterative Probing and Score Comparison:** The adversary performs a series of similarity computations. They query with their vector $\mathbf{x}$ against subsets of the database belonging to different creators. For each query against an encrypted vector $Enc(\mathbf{y}_{artist\_A})$ from Artist A's collection, they obtain a similarity score $s_A = \mathbf{x} \cdot \mathbf{y}_{artist\_A}$ (after decryption by a key-holding entity). They repeat this process for Artist B, Artist C, and so on.

4. **Inference via Score Discrepancy:** The information leak occurs by comparing the resulting similarity scores. If the score s_A is consistently and significantly higher than the scores obtained from other artists' collections $(s_B, s_C, \dots)$, it creates a strong statistical link. The adversary can reasonably infer that the disputed track $\mathbf{x}$ was most likely derived from, or heavily inspired by, the original work of Artist A.

This form of attack poses a serious threat to creator privacy and intellectual property. It could be exploited in various ways: a malicious actor could attempt to falsely attribute a low-quality track to a famous artist to cause reputational damage; a party in a copyright dispute could use this method to gather evidence; or a collaborator could check if their creative partner is re-using their protected, encrypted contributions in unauthorized projects. This highlights the need for privacy-preserving mechanisms that protect not only the content but also the metadata and context surrounding the creator's identity.

4.2 Efficient Homomorphic Inner Product for Music Embedding Vectors

4.2.1 Blocked Inner Product for Structural Vectors

A standard inner product treats a high-dimensional vector as a monolithic entity, which is often insufficient for the nuanced requirements of music similarity search. Music embeddings are not merely flat lists of numbers; they possess a rich internal structure designed to capture music's complex and layered nature. For instance, a single embedding vector might contain distinct, contiguous segments representing rhythmic patterns, melodic contours, harmonic progressions, and timbral qualities. A simple inner product would conflate these distinct features, potentially allowing high similarity in one aspect (e.g., rhythm) to be diluted by dissimilarity in another (e.g., melody), leading to less relevant search results.

To address this challenge and leverage the inherent structure of music embeddings, we propose a *Blocked Inner Product* approach. In this scheme, we partition a high-dimensional plaintext query vector $\mathbf{x}$ and its corresponding encrypted database vector $Enc(\mathbf{y})$ into k semantically meaningful blocks:

$$\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\} \quad \text{and} \quad \mathbf{y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}.$$

Each block $\mathbf{x}_i$ and $\mathbf{y}_i$ corresponds to a specific musical feature. The encrypted similarity is then computed as the homomorphic sum of the inner products of these corresponding blocks:

$$Sim_{blocked}(\mathbf{x}, Enc(\mathbf{y})) = \bigoplus_{i=1}^{k} Sim(\mathbf{x}_i, Enc(\mathbf{y}_i)) = Enc\left(\sum_{i=1}^{k} \mathbf{x}_i \cdot \mathbf{y}_i\right), \quad (1)$$

where $Enc(\mathbf{y}_i)$ represents the encryption of the i-th block of vector $\mathbf{y}$. This approach allows for more granular similarity comparisons. By isolating

features into blocks, our method prevents the "averaging out" of important musical characteristics and provides a more interpretable and structurally aware measure of similarity. Furthermore, this partitioning creates the necessary foundation for the weighted retrieval scheme discussed in the following section, where different levels of importance can be assigned to each musical aspect.

4.2.2 Weighted Hierarchical Inner Product

While the blocked inner product provides a more structured comparison, it still assumes that each musical feature block is equally important for any given search. This one-size-fits-all approach fails to capture the task-specific nature of music retrieval. For instance, a query to find songs with a "similar groove" should prioritize rhythmic and bassline features, whereas a query for "lyrically similar" songs would focus on blocks representing vocal melody and ignore instrumentation. The ability to dynamically emphasize or de-emphasize certain features is crucial for building a truly intelligent and flexible retrieval system.

To achieve this, we extend the blocked product to a *Weighted Hierarchical Inner Product*. This method assigns a public, plaintext weight, w_i, to the similarity score of each block. These weights are not fixed; they are chosen based on the specific retrieval task, allowing the system to adapt its definition of similarity on a per-query basis.

The weighted hierarchical similarity is computed as:

$$\begin{aligned} Sim_{weighted}(\mathbf{x}, Enc(\mathbf{y})) &= \bigoplus_{i=1}^{k} Sim(\mathbf{x}_i, Enc(\mathbf{y}_i))^{w_i} \\ &= Enc\left(\sum_{i=1}^{k} w_i \cdot (\mathbf{x}_i \cdot \mathbf{y}_i)\right), \end{aligned} \tag{2}$$

where the weights w_i are public parameters chosen based on the retrieval task. This method provides enhanced flexibility without compromising the underlying additive homomorphic properties. The application of weights is performed as an efficient plaintext-ciphertext multiplication, which is natively supported by the AHE scheme and avoids the costly operations associated with FHE.

This weighted approach offers significant advantages. It transforms the similarity search from a static operation into a dynamic, context-aware process. A system can now support diverse query types—from finding songs with a similar instrumentation to identifying tracks with a specific mood—by

simply adjusting the weight vector $\{w_1, \ldots, w_k\}$. This provides a powerful mechanism for building sophisticated, privacy-preserving music search and recommendation engines that are both efficient and highly adaptable to user intent.

5 Evaluation

5.1 Experimental Setup

The experiment is carried out with an AWS EC2 t3.xlarge VM. Music data source is retrieved from the MagnaTagATune dataset [26] and we sampled 1,000 MP3 files from the dataset for music embeddings, each mapped as a vector (plaintext or encrypted) in Milvus [11]. We use YAMNet [27] to generate music embeddings with different lengths, i.e., 128, 256, 512, and 1024. We employed Microsoft TenSEAL [28] to apply encryption schemes including FHE and AHE. For AHE, we consider the below two different settings.

Encrypted Database Setting: in this scenario, the database vectors are encrypted, and the query vector remains plaintext. Here, the dot product is taken similarly: each encrypted database value is added to itself x_i times, with x_i coming from the plaintext query.

Encrypted Query Setting: in this variation, only the query vector is encrypted. The database is in plaintext. Instead of using multiplication, the dot product is computed through fast repeated addition, where each encrypted query element is added to itself y_i times (with y_i being the corresponding value in the plaintext database vector). This avoids ciphertext–ciphertext operations and uses the additive function. This method reduces memory usage and runtime avoiding encrypting large database vectors and operates well ciphertext–plaintext interactions.

5.2 System Implementation

We implemented vector dot product computations using both AHE and FHE schemes across a range of embedding vector lengths (1024, 512, 256, and 128). These dimensions reflect typical sizes used in music embedding systems such as those based on CLAP [29] or wav2vec [30]. The comparison is conducted between FHE and AHE under both encrypted query and encrypted database settings that are defined in the experimental setup section.

The FHE implementation takes TenSEAL's CKKS scheme to encrypt both the query vector and the database vectors. Each dot product is computed

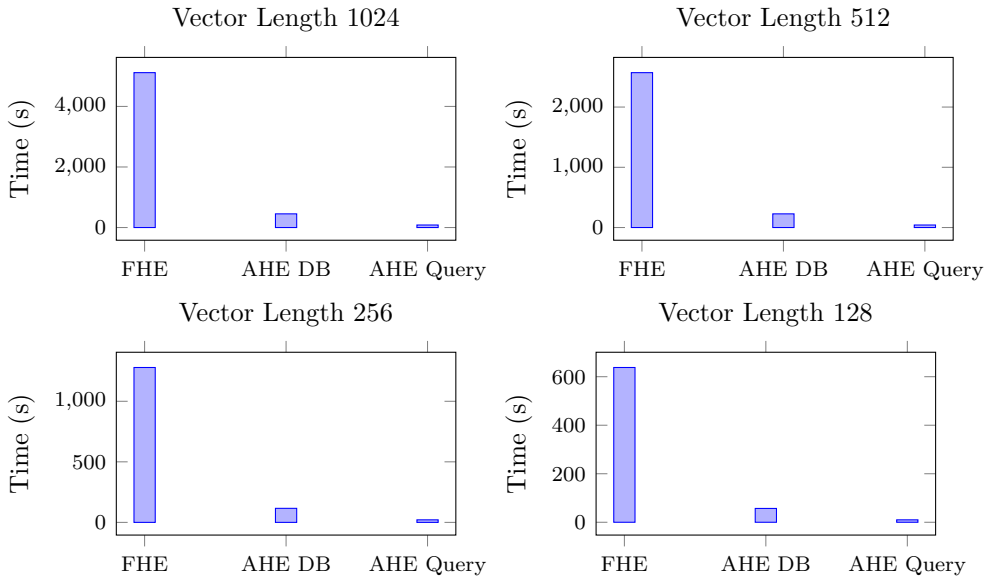

Figure 1: Dot product computation times using FHE and AHE methods for different vector lengths.

through: Performing ciphertext–ciphertext multiplication for each element, then summing the encrypted values. This supports real numbers and allows arbitrary computations on encrypted data but causes a longer runtime and significant memory use. The encryption context is relatively heavy, and homomorphic multiplication increases noise.

The AHE implementation uses the same TenSEAL's CKKS scheme to encrypt either the query vector or the database vectors under the two different settings in the proposed approach, i.e., Encrypted Query Setting and Encrypted Database Setting.

5.3 Preliminary Results

5.3.1 Baseline Comparison

As shown in Figure 1, the AHE implementations completed the dot product operations substantially faster in all dimensions, highlighting its viability in handling the encryption/description of embeddings much more efficiently, which is an essential requirement in music IR contexts. The encrypted query setting, in particular, showed the best performance since it minimizes the number of encrypted values processed, aligning well with real-world

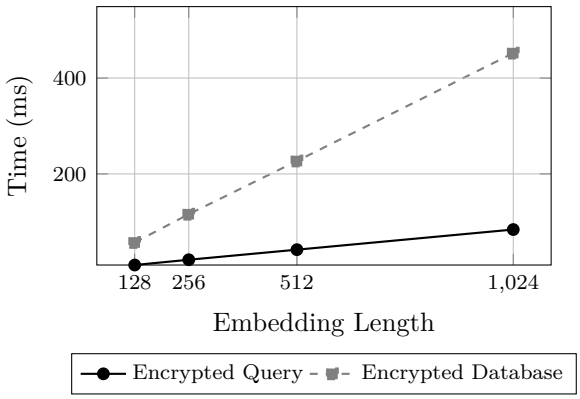

Figure 2: Trend of AHE-based dot product runtime across different embedding lengths.

client–server setups where only the user's input (query) needs protection.

5.3.2 Scalability

The ability to handle vectors of varying lengths is crucial for the flexibility of music IR applications, enabling them to capture different levels of information in embeddings. In Figure 2, we demonstrate that AHE achieves linear computational time with respect to vector size (i.e., 128, 256, 512, and 1024), applicable to both encrypted database and encrypted query settings. This efficiency is due to the reduced ciphertext handling in AHE. Notably, in the encrypted query setting, the server's workload closely mirrors that of a plaintext dot product.

5.3.3 Memory footprint

We also tested the memory usage across the FHE and AHE-based methods on 1024-length vectors as shown in Figure 3. Based on this experiment, AHE approach under Encrypted Database Setting (i.e., AHE DB in the figure) exhibits similar memory consumption compared to FHE, while AHE approach under Encrypted Query Setting (i.e., AHE Query in the figure) shows significantly lower memory. This is because both FHE and AHE DB methods require storing and processing large ciphertexts that encode entire vectors, significantly inflating memory usage due to ciphertext expansion. These ciphertexts often include metadata for noise management, modulus switching, and require more complex data structures for intermediate computations.

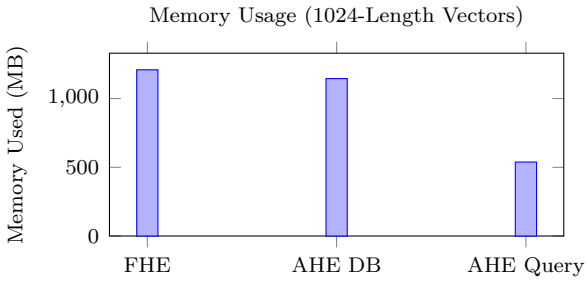

Figure 3: Memory usage for dot product operations on 1024-length vectors using FHE and AHE schemes.

6 Ongoing Efforts

6.1 Lifecycle Management for Semantically-Aware Music Databases

The ultimate vision is to create a full-fledged Database Management System (DBMS) for encrypted, semantically-structured vector data. This system would move far beyond a simple cryptographic library, offering a comprehensive suite of features analogous to modern relational DBMSs. This includes a declarative query language for expressing complex similarity criteria, a cost-based query optimizer that understands the semantics of encrypted data structures, and robust protocols for multi-tenant data management, schema evolution, and transactional updates. The goal is to make the management of sensitive, high-dimensional AI data as tractable, flexible, and secure as traditional enterprise data, enabling a new generation of privacy-preserving intelligent applications.

To achieve the above goal, this foundational paper introduces new concepts of a *Blocked Inner Product* and a *Weighted Hierarchical Inner Product* to embed musical structure into similarity computations. A direct extension is to build a complete system around these static, query-time concepts. This involves first formalizing a data model and a declarative query interface (e.g., a SQL extension) that allows users to specify blocks and weights dynamically. Subsequent research must address the full data lifecycle: developing efficient and atomic protocols for updating a vector's semantic block structure or associated metadata without requiring full re-encryption. Furthermore, a robust key management framework is necessary to handle multi-owner scenarios, where different rights-holders (e.g., music labels) contribute to a single, shared database while retaining control over their assets.

6.2 Federated Music Retrieval Systems

The vision is a secure and decentralized marketplace for creative AI assets, often described as a "data clean room" for vector embeddings. In such an ecosystem, multiple, mutually distrusting parties could securely query each other's proprietary databases. This would enable critical business functions including copyright-infringement checks, content licensing, and trend analysis, all without any party revealing its private query data or its confidential database content to the other. This paradigm shifts from a client-server trust model to a federated, multi-party trust environment, unlocking collaborative potential while enforcing privacy for all participants.

To achieve the above goal, this foundational paper effectively analyzes threat models like Melody Inference and Creator Identity Inference, proposing AHE-based solutions for scenarios where either the query or the database is encrypted. The critical next step is to address the mutual privacy scenario, where both the query `Enc(x)` and the database `Enc(y)` must be protected from each other. As the paper notes, FHE is too inefficient for this task. The extension would involve designing and evaluating a novel hybrid system architecture. A promising approach combines the efficiency of the paper's AHE scheme with the power of Trusted Execution Environments (TEEs) for computation. In this model, AHE-encrypted data would be securely loaded into a TEE, decrypted for high-speed plaintext computation, and the results re-encrypted before exiting, ensuring that even the server administrator cannot access the sensitive data. This requires formalizing the multi-party threat model and evaluating the performance and security trade-offs.

7 Conclusion

This paper demonstrates that Additive Homomorphic Encryption (AHE) offers a practical and efficient solution for privacy-preserving music information retrieval, striking a crucial balance between security and performance. We have analyzed threat models unique to music data, proposed structure-aware inner product methods to enable meaningful similarity searches on encrypted vectors, and empirically verified that our AHE-based approach is substantially faster and more memory-efficient than traditional FHE. Our ongoing work builds upon this foundation, aiming to develop (i) full-fledged database management systems for semantically-aware music data and (ii) federated platforms for mutually private collaboration of music works in the creative AI ecosystem.

References

[1] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *OpenAI Blog*, 2020. `https://openai.com/blog/jukebox`.

[2] Andrea Agostinelli, Matthieu Caron, Theo Copet, Alexandre Défossez, Jack Frankel, Kunal Goel, Balázs Hidasi, Antoine Liutkus, Matthieu Monfort, Olivier Pietquin, et al. Musiclm: Generating music from text. *arXiv preprint arXiv:2301.11325*, 2023.

[3] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. Deep learning techniques for music generation–a survey. *arXiv preprint arXiv:1709.01620*, 2017.

[4] Philippe Esling, Axel Chemla-Romeu-Santos, and Jordan Bitton. Bridging audio analysis, synthesis, and processing with deep learning: A survey. *IEEE Transactions on Audio, Speech, and Language Processing*, 26(12):1–13, 2018.

[5] Samuel Ferguson. Generative ai and the music industry: Threats and opportunities. `https://jolt.law.harvard.edu/digest/generative-ai-and-the-music-industry-threats-and-opportunities`, 2023. Harvard Journal of Law & Technology Digest.

[6] Ben Sisario. A.i. song featuring fake drake and the weeknd goes viral — and gets pulled. *The New York Times*, April 2023.

[7] Diya Zhang. Studio ghibli fans push back against a.i.-generated art. *The Verge*, July 2023.

[8] Josiah Hester. Shazam it! music processing, fingerprinting, and recognition. *Toptal Engineering Blog*, 2023.

[9] Lynkz. What are vector databases? understanding the 2024 landscape. *Lynkz Blog*, 2024.

[10] Solmaz Seyed Monir and Dongfang Zhao. Efficient feature extraction for image analysis through adaptive caching in vector databases. In *2024 7th International Conference on Information and Computer Technologies (ICICT)*, pages 193–198, 2024.

[11] Jianguo Wang et al. Milvus: A purpose-built vector data management system. In *SIGMOD*, pages 2614–2627, 2021.

[12] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing (STOC)*, 2009.

[13] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *23rd International Conference on the Theory and Applications of Cryptology and Information Security (AsiaCrypt)*. Springer, 2017.

[14] Yin Li, Dhrubajyoti Ghosh, Peeyush Gupta, Sharad Mehrotra, Nisha Panwar, and Shantanu Sharma. Prism: Private verifiable set computation over multi-owner outsourced databases. In *Proceedings of the 2021 International Conference on Management of Data*, SIGMOD '21, page 1116–1128, New York, NY, USA, 2021. Association for Computing Machinery.

[15] Olamide T. Tawose, Jun Dai, Lei Yang, and Dongfang Zhao. Toward efficient homomorphic encryption for outsourced databases through parallel caching. *Proceedings of the ACM on Management of Data (SIGMOD)*, May 2023.

[16] Pinecone: A vector database for machine learning. `https://www.pinecone.io/`, 2025.

[17] Yu A Malkov and D A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2020.

[18] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. In *IEEE Trans. Big Data*, pages 535–547, 2019.

[19] Cheng Chen, Chenzhe Jin, Yunan Zhang, Sasha Podolsky, Chun Wu, Szu-Po Wang, Eric Hanson, Zhou Sun, Robert Walzer, and Jianguo Wang. Singlestore-v: An integrated vector database system in singlestore. *Proc. VLDB Endow.*, 17(12):3772–3785, August 2024.

[20] Dongfang Zhao. A note on efficient privacy-preserving similarity search for encrypted vectors. *arXiv preprint arXiv:2502.14291*, 2025.

[21] Yi Ren, Jinglin Liu, Tong Zou, Zhou Liu, Zhijie Zhao, and Zhiyao Zhao. Pirhdy: Learning pitch, rhythm, and dynamics representations for symbolic music. *arXiv preprint arXiv:2010.08091*, 2020.

[22] Wenjing Yang, Keunwoo Choi, Xavier Serra, Xubo Zhang, and Juho Nam. Towards robust multimodal music understanding via supervised contrastive learning. *arXiv preprint arXiv:2404.13569*, 2024.

[23] Cory McAllister, Igor Aizenberg, and Douglas Turnbull. Cover detection with contrastive learning: Feasibility and shortcomings. *arXiv preprint arXiv:2109.02472*, 2021.

[24] Christian Fremerey and Meinard Müller. Music retrieval evaluation: From ad-hoc retrieval to standardized test collections and beyond. *Dagstuhl Follow-Ups*, 3:93–118, 2013.

[25] Dev Aggarwal. Shazam's algorithm. `https://medium.com/@dream-y/shazams-algorithm-5ba2b0a60d7a`, 2022. Accessed: 2025-07-26.

[26] Edith Law, Olivier Gillet, and J. Stephen Downie. The magnatagatune dataset. `https://mirg.city.ac.uk/codeapps/the-magnatagatune-dataset`, 2009. City, University of London.

[27] Manoj Plakal and Daniel P. W. Ellis. Yamnet: A deep net that predicts 521 audio event classes from the audioset-youtube corpus. `https://www.tensorflow.org/hub/tutorials/yamnet`, 2020. TensorFlow Hub.

[28] Ayoub Benaissa, Bilal Retiat, Bogdan Cebere, and Alaa Eddine Belfedhal. Tenseal: A library for encrypted tensor operations using homomorphic encryption, 2021.

[29] Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Marianna Nezhurina, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation, 2024.

[30] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations, 2020.