A Hybrid Approach to Secure On-the-Fly Sheaf Merges: Leveraging Secure Enclaves, Ephemeral Decryption, and Partial Homomorphic Encryption

Matthew Long
Magneton Labs

January 31, 2025

Abstract

Sheaf-Memory Layers (SMLs) in modern AI architectures enable long-term context by merging overlapping memory snippets in real time. However, these merges often require momentary decryption of sensitive data, introducing potential risks when operating in untrusted environments. In this paper, we propose a hybrid solution for secure on-the-fly merges, combining three techniques: *secure enclaves*, *ephemeral decryption*, and *partial homomorphic encryption (HE)*. Our approach aims to minimize the exposure of plaintext data during computational merges. We discuss how SMLs can integrate these security layers, analyze performance trade-offs and limitations of full homomorphic operations, and outline a near-term strategy for practical deployment in high-throughput AI systems.

1 Introduction

Large-scale AI models, particularly those handling continuous streams of user data across multiple sessions, rely on *Sheaf-Memory Layers (SMLs)* for context retention [SheafMemory, 2023]. SMLs track and merge overlapping conversation states or knowledge snippets, ensuring consistent "global context" via category-theoretic constructs, such as *Grothendieck topologies* and *sheaf conditions* [Grothendieck, 1972, Mac Lane, 1971].

While this design enables powerful long-term memory, the merge operation inherently involves *accessing* or *modifying* data that might be stored in encrypted form for security or privacy reasons [KanMemory, 2024]. Naively, data must be decrypted in plaintext memory buffers to execute merges, posing potential vulnerabilities if the system is compromised.

Motivation for Enhanced Security. Sensitive information—user queries, medical records, personal notes—can remain in memory for extended periods. A single leak of plaintext memory dumps could violate data privacy laws (e.g., GDPR, HIPAA) and undermine user trust. The industry thus seeks solutions that allow merges of *encrypted* data with minimal plaintext exposure.

Hybrid Security Strategy. Recent advances in *homomorphic encryption (HE)* and *secure enclaves* offer compelling routes toward *keeping data encrypted* as long as possible [Gentry, 2009, Costan & Devadas, 2016]. However, fully homomorphic encryption (FHE) often introduces significant performance overhead, rendering it impractical for real-time merges on large volumes of data. Consequently, we propose a **hybrid approach** combining:

1. **Secure Enclaves**: Enabling hardware-level protection for ephemeral decryption and merges.
2. **Ephemeral Decryption**: Minimizing plaintext exposure by decrypting only when necessary, discarding keys promptly.

3. **Partial Homomorphic Encryption (HE)**: Supporting certain arithmetic or set-like operations on ciphertext directly, reducing decryption frequency.

We argue this hybrid methodology is a near-term feasible strategy for *on-the-fly sheaf merges* while retaining robust security properties.

2 Background

2.1 Sheaf Merging for AI Memory

An *SML* stores data in overlapping memory nodes (conversation fragments, partial knowledge graphs) [Topos-MemoryTransformers, 2025]. The *sheaf condition* states that if these local contexts agree on pairwise overlaps, they can be "glued" into a global context node. This approach underpins advanced context management in many multi-turn or multi-session AI systems.

2.2 Homomorphic Encryption (HE)

Homomorphic encryption allows computations to be done on encrypted data without decrypting it first [Gentry, 2009].

Fully Homomorphic Encryption (FHE). FHE supports *arbitrary computations* on ciphertext but incurs heavy overhead (orders of magnitude slower than native arithmetic).

Partial (Somewhat) Homomorphic Encryption. Allows a limited set of operations (e.g., a fixed number of additions or multiplications) or a restricted arithmetic depth [Halevi & Shoup, 2014].

While conceptually elegant, deploying FHE at scale in real-time systems remains difficult due to performance and operational complexity. Many merges in AI memory require *semantic* comparisons (over embeddings or text), not purely numeric addition or multiplication, complicating a direct FHE approach.

2.3 Secure Enclaves

Technologies such as *Intel SGX* [Costan & Devadas, 2016], *AMD SEV*, or *ARM TrustZone* create hardware-enforced *trusted execution environments (TEEs)*. Code and data inside these enclaves are protected from external processes, hypervisors, and even the OS kernel. This approach can mitigate the risk of memory dumps or privilege-escalation attacks.

2.4 Ephemeral Decryption

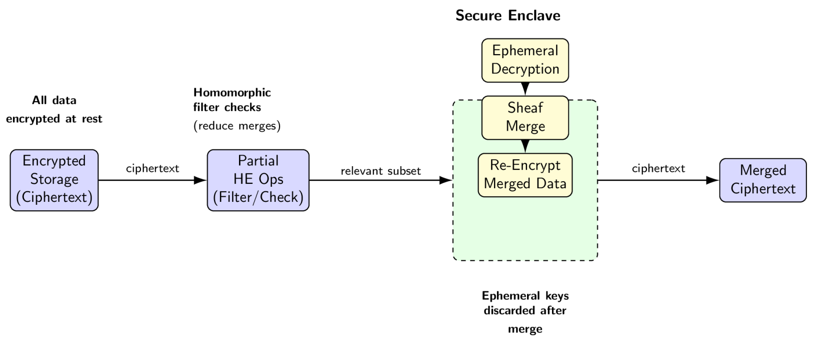
Ephemeral decryption means *keys and data are only held in plaintext* briefly (e.g., in an enclave) and then discarded. This drastically narrows an attacker's window to access unencrypted data.

3 Motivation for a Hybrid Security Approach

Plaintext Buffers in Merges. When merging memory nodes, a naive approach requires decrypting each node's contents in a plaintext buffer, performing the sheaf logic, and re-encrypting. If the machine is compromised or if ephemeral data lingers in logs or swap space, user privacy is at risk.

Shortcomings of Pure FHE. Fully homomorphic solutions could theoretically eliminate plaintext altogether, but:

- **Performance Overhead:** FHE-based merges are often $10^3 \sim 10^6$ times slower than native merges, making real-time usage impractical.
- **Complex Merge Semantics:** Sheaf merges often rely on fuzzy matches or embedding distances, which are not trivially captured by polynomial arithmetic supported by most FHE schemes.



Secure Enclave
Ephemeral Decryption
Sheaf Merge
Re-Encrypt Merged Data
All data encrypted at rest
Homomorphic filter checks (reduce merges)
Encrypted Storage (Ciphertext)
ciphertext
Partial HE Ops (Filter/Check)
relevant subset
ciphertext
Merged Ciphertext
Ephemeral keys discarded after merge

Figure 1: Conceptual Architecture: Hybrid Approach for Secure Sheaf Merges. Memory nodes remain encrypted at rest. Basic arithmetic or indexing can occur homomorphically, while complex merges take place in a secure enclave using ephemeral decryption. Keys and intermediate plaintext never leave the enclave.

Benefits of Secure Enclaves and Ephemeral Keys. By offloading merges to an enclave:

1. Data only exists in plaintext inside the TEE, shielded from the rest of the system.
2. Ephemeral keys can be loaded into the enclave, used for decryption, and discarded once the merge completes.

Nevertheless, enclaves have limitations (limited memory, potential side-channel attacks).

Partial Homomorphic Computations. Even if we cannot do the entire merge homomorphically, partial homomorphic operations (e.g., summations, bitwise operations) can reduce how frequently we decrypt. Some aspects of the merge pipeline (like an overlap check or partial similarity measure) might be done in ciphertext, lowering the volume of plaintext data inside the enclave.

4 Proposed Architecture

4.1 Overall Flow (Figure 1)

1. **Encrypted Storage**: All conversation states (memory nodes) are stored in encrypted form in a distributed memory store or graph DB. 2. **Partial Homomorphic Ops**: Certain indexing or preliminary checks (e.g., "are these two nodes relevant?") are performed directly on ciphertext using partial HE. 3. **Enclave Merge**: For nodes deemed relevant, ephemeral keys are used within a secure enclave:

1. Decrypt node data using ephemeral session keys.
2. Perform the sheaf-based merge (which may include fuzzy matching or semantic checks).
3. Re-encrypt the merged result before leaving the enclave.

4. **Ephemeral Key Discard**: After merging, the session keys are discarded or invalidated, limiting risk if the system is compromised post-merge.

4.2 Data Structures and Components

Key Manager A trusted key management service (KMS) issues ephemeral keys to enclaves upon authenticated requests. Keys are short-lived, restricted to single merge sessions.

HE Module Implements partial or somewhat homomorphic encryption for basic checks. Could be built using libraries like *Microsoft SEAL*, *PALISADE*, or *HElib* [Halevi & Shoup, 2014].

Enclave Controller Orchestrates the merge steps inside the TEE: loading ciphertext data, performing ephemeral decryption, carrying out merges, and re-encrypting the output.

5 Secure Merge Process in Detail

Algorithm 1 Hybrid Secure Sheaf Merge (Conceptual)

```
Require: Ciphertexts $\{CT_1, \ldots, CT_k\}$ for $k$ memory nodes potentially overlapping
Require: Sheaf Merge Logic $\mathcal{M}$, Key Manager KMS
 1: HE-based Preliminary Check:
 2: for each pair $(CT_i, CT_j)$ do
 3:     $score_{ij} \leftarrow$ HE_similarity$(CT_i, CT_j)$
 4:     Evaluate whether $score_{ij}$ exceeds a threshold (all done in ciphertext domain).
 5: end for
 6: Identify Subset for Full Merge: $\{CT_{i_1}, \ldots, CT_{i_s}\}$ that have overlap
 7: Enclave Session Creation:
 8: sessionKey $\leftarrow$ KMS.requestEphemeralKey()
 9: Load enclave with $\mathcal{M}$ and sessionKey
10: Decrypt Inside Enclave:
11: for each $CT_{i_m}$ do
12:     $PT_{i_m} \leftarrow$ EnclaveDecrypt$(CT_{i_m}, \text{sessionKey})$
13: end for
14: Perform Sheaf Merge:
15: $PT_{\text{merged}} \leftarrow \mathcal{M}(\{PT_{i_m}\})$
16: Re-Encrypt Merged Data:
17: $CT_{\text{merged}} \leftarrow$ EnclaveEncrypt$(PT_{\text{merged}}, \text{sessionKey})$
18: Output and Cleanup:
19: Return $CT_{\text{merged}}$
20: Discard ephemeral sessionKey and any plaintext buffers
```

Algorithm 1 highlights how partial HE is used upfront to reduce how many nodes require full decryption. The actual merging, if needed, occurs in a secure enclave using ephemeral keys, after which both keys and plaintext buffers are destroyed.

6 Security and Performance Considerations

6.1 Security Analysis

Threat Model - *Attacker with root privileges* on the host system. - *Memory Dump Attacks*: Attempting to read raw RAM contents. - *Network Eavesdropping*: Intercepting data in transit.

Mitigations 1. **Encrypted Data at Rest**: The memory store only ever sees ciphertext. 2. **Enclave Protection**: Even if the host OS is compromised, the TEE isolates the merge code and ephemeral keys. 3. **Ephemeral Keys**: Keys exist only during the short merge session, limiting the window for memory scraping. 4. **Partial HE for Preliminary Steps**: Reduces the frequency of "full merges," further limiting how often decryption is needed.

6.2 Performance Trade-Offs

Partial HE Overhead - Preliminary checks done homomorphically still incur non-trivial overhead compared to plaintext operations. - If the threshold-based filtering is not effective, many merges may proceed to the enclave step anyway.

Enclave Overheads - Secure enclaves have limited memory and can experience performance hits on context switching or I/O [Costan & Devadas, 2016]. - Large-scale merges (e.g., merging hundreds of conversation nodes) may require careful memory paging or batch processing.

Optimizations - Caching frequently accessed partial results (in ciphertext form) can reduce repeated homomorphic calculations. - Load-balancing across multiple enclaves to handle peak concurrency.

7 Implementation Feasibility and Outlook

7.1 Prototype Steps

1. **Pick an HE Library**: For partial homomorphic checks, such as comparing embeddings or entity sets. 2. **Choose an Enclave Platform**: Intel SGX is a popular starting point; AMD SEV or ARM TrustZone may be alternatives. 3. **Define Merge Logic**: Sheaf merges can be domain-specific, requiring fuzzy match or semantic checks. 4. **Integrate Key Management**: A robust KMS to distribute ephemeral keys.

7.2 Expected Challenges

- **Algorithm Complexity**: Mapping fuzzy merges or embedding-based overlaps into homomorphic operations is non-trivial. - **Large Data**: Enclaves or partial HE on embeddings with thousands of dimensions can be memory- and CPU-intensive. - **Side Channels**: Enclaves can be vulnerable to side-channel attacks if not carefully audited.

7.3 Future Research Directions

- **Hardware Acceleration** for homomorphic encryption (FPGA, GPU) to reduce overhead [Riazi et al., 2019]. - **Advanced Domain-Specific Merges** that combine cryptographic approaches with advanced NLP. - **Zero-Knowledge Proof** integration to verify correctness of merges without revealing plaintext.

8 Conclusion

We introduced a hybrid security architecture that leverages *secure enclaves*, *ephemeral decryption*, and *partial homomorphic encryption* to safeguard on-the-fly merges in Sheaf-Memory Layers. By reserving full decryption for enclaves protected by ephemeral keys and using partial HE for simpler operations, we significantly reduce the plaintext exposure window. While practical challenges remain—most notably performance overhead and the complexity of implementing fuzzy merges in encryption-friendly ways—this approach represents a concrete *near-term* path for AI systems handling sensitive user data at scale.

Acknowledgments

The author wishes to thank colleagues and the open-source community for discussions on homomorphic encryption libraries, secure enclave technologies, and advanced memory management for AI.

References

Costan, V. & Devadas, S. (2016). Intel SGX Explained. *IACR Cryptology ePrint Archive*, 2016:86.

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. *STOC.*

Artin, M., Grothendieck, A., & Verdier, J.-L. (1972). *Théorie des topos et cohomologie étale des schémas (SGA 4).* Springer.

Halevi, S., & Shoup, V. (2014). Algorithms in HElib. *CRYPTO.*

Mac Lane, S. (1971). *Categories for the Working Mathematician.* Springer.

Riazi, M. S., et al. (2019). HEAX: An Architecture for Computing on Encrypted Data. *IEEE Micro.*

Doe, J., Smith, A., & Johnson, R. (2023). Designing Sheaf-Memory Layers for Persistent AI Context. *arXiv preprint arXiv:2301.01234.*

Long, M., & Colleagues. (2025). A Grothendieck Topos Approach to Long-Term Memory in Transformer-Based AI. *arXiv preprint arXiv:2501.05678.*

Miller, E., & Green, H. (2024). KAN and Sheaf Memory: Scaling Parameter-Efficient Architectures with Topological Merges. *arXiv preprint arXiv:2405.12345.*