DM-TEE: Trusted Execution Environment for Disaggregated Memory

Ke Xia
ke.xia@rutgers.edu
Rutgers University
Piscataway, NJ, USA

Sheng Wei
sheng.wei@rutgers.edu
Rutgers University
Piscataway, NJ, USA

ABSTRACT

Trusted execution environments (TEEs) can provide hardware and system-level protection for sensitive data and computations. However, the security perimeter of existing TEEs is limited to a single centralized machine, which contradicts with the growing trend of employing disaggregated computing resources (e.g., disaggregated memory) to achieve high performance and resource utilization. To address this limitation, we develop *DM-TEE*, a customized trusted execution environment supporting the emerging disaggregated memory architecture. *DM-TEE* extends the traditional TEEs from local memory to remote disaggregated memory, which is achieved by a newly designed secure memory allocation and access workflow to ensure the data confidentiality and integrity in the disaggregated memory. We implement *DM-TEE* on real hardware using Intel SGX and a state-of-the-art memory disaggregation system. Our evaluations on memory allocation, read/write operations, and benchmark program executions indicate that *DM-TEE* achieves the desired disaggregated memory security with minimal performance overhead.

CCS CONCEPTS

• **Security and privacy** → **Trusted computing**; **Security in hardware**.

KEYWORDS

Trusted Execution Environment, Disaggregated Memory

ACM Reference Format:
Ke Xia and Sheng Wei. 2024. *DM-TEE*: Trusted Execution Environment for Disaggregated Memory. In *Great Lakes Symposium on VLSI 2024 (GLSVLSI '24), June 12–14, 2024, Clearwater, FL, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649476.3658702

1 INTRODUCTION

Trusted execution environments (TEEs), such as ARM TrustZone [1] for mobile devices and Intel SGX [8] for server platforms, have drawn a great deal of attention recently providing hardware/system-level protection for sensitive data and computations. Compared to its predecessor isolation approaches at the OS and hypervisor levels [15, 17], the hardware isolation provided by TEE ensures the confidentiality and integrity of the data even if the OS kernel has been compromised, essentially excluding the huge-size OS from the trusted computing base (TCB) to achieve the minimum attack surface and thus highest level of security guarantee. In the past decade, TEEs have become the state-of-the-art system security mechanism and demonstrated strong potentials in a large number of applications, such as mobile computing [16], OS kernels [20], and cloud computing [3].

Despite the strong security guarantee, the existing design of TEEs has a fundamental limitation that its security perimeter is bounded to a single CPU core on a centralized machine, and all the security-sensitive data and computations must be deployed in a resource-constrained container (e.g., *secure world* in TrustZone or *enclave* in SGX). Unfortunately, this limitation contradicts with the growing trend of employing distributed computing resources to achieve high performance and resource utilization. For example, the emerging *disaggregated memory architecture* [6, 12, 14] incorporates physically distributed memory devices residing in a remote memory pool to expand the resource capacity of the local computing node and improve the resource utilization and flexibility. However, such disaggregated memory systems cannot be secured by the state-of-the-art TEEs due to the aforementioned limitation.

There are several recent research efforts in the community aiming to extend the singular TEEs to peripheral devices [24], GPUs [21], and FPGAs [25] by building an encryption-based secure path. Also, several works proposed to build standalone TEEs that support heterogeneous computing resources [7, 27]. Despite their effectiveness in certain target application scenarios, the existing approaches are still under the scope of singular TEE physically located in a single computing node, which cannot be applied to protecting the emerging disaggregated memory architecture.

We develop *DM-TEE*, a customized trusted execution environment to protect disaggregated memory systems. *DM-TEE* extends the traditional TEE from local memory to remote disaggregated memory, supporting secure disaggregated memory allocation and access workflow. On the client side, *DM-TEE* employs an enclave memory (eMem) monitor to generate the remote memory request from the original enclave, as well as maintain a customized memory integrity tree based on the mountable Merkle tree (MMT) structure [10] to reduce the time complexity of runtime integrity check. On the remote disaggregated memory side, *DM-TEE* deploys a secured memory node (SMN) to manage the disaggregated DRAM nodes and perform permission check by employing the EPCM structure [8]. Furthermore, the client and the disaggregated memory are bridged by a remote memory (rMem) manager, which conducts a customized attestation process to build mutual trust between the local and remote memories.

GLSVLSI '24, June 12–14, 2024, Clearwater, FL, USA

ACM ISBN 979-8-4007-0605-9/24/06
https://doi.org/10.1145/3649476.3658702

We implement *DM-TEE* on real hardware using the state-of-the-art Intel SGX [8] as the TEE and Clio [12] as the memory disaggregation system. Our evaluations indicate that *DM-TEE* achieves the desired disaggregated memory security with minimal performance overhead. To the best of our knowledge, *DM-TEE* is the first TEE that supports the disaggregated memory architecture, which extends the memory capacity and utilization of TEEs by enabling remote memory accesses while maintaining the same security guarantee.

2 BACKGROUND AND RELATED WORKS

2.1 Disaggregated Memory

Memory disaggregation aims to decouple memory resources from traditional centralized computer architecture and allows data to be stored in a remote memory pool. Most disaggregated memory solutions deploy RDMA to enable remote memory access [11, 22], which provides low average latency and high throughput. However, RDMA introduces a large overhead with the large number of page table entries (PTEs). To address the known issues, recent memory disaggregation systems like Clio [12] employ customized hardware to facilitate the high-speed communications, manage the large memory capacity, and handle the large number of of concurrent requests. In particular, Clio constructs three distinct paths to optimize the performance of disaggregated memory systems [12]. The slow path in the ARM processor handles metadata operations, including the memory allocation for both the virtual address (VA) and the physical address (PA). The fast path in ASIC performs memory access, including address translation, permission check, and page fault handling. The customized memory requests are handled by the extend path.

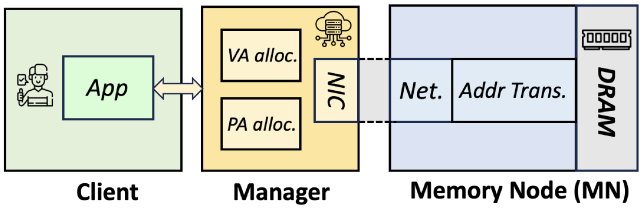


Figure 1: Overview of disaggregated memory system.

Without loss of generality, we implemented a prototype disaggregated memory system based on Clio [12] for the research of *DM-TEE*, as illustrated in Figure 1. The client can communicate with the manager to request the memory node (MN). The manager processes the memory allocation requests (i.e., the slow path), and the MN performs the address translation to access the DRAM (i.e., the fast path). Note that the existing disaggregated memory designs do not consider the potential security threats in the disaggregated architecture, which is addressed by *DM-TEE*.

2.2 Trusted Execution Environment (TEE)

Trusted Execution Environments (TEEs) are designed to isolate the security-sensitive computation from untrusted applications and protect the data confidentiality and integrity. The state-of-the-art TEEs, such as SGX [8], provide comprehensive protection for the enclave data and code. In SGX [8], the enclave is stored within a protected memory region known as the Processor Reserved Memory (PRM), and the enclave page cache (EPC) is a subpart of PRM, which is inaccessible to the operating system and untrusted software. Furthermore, SGX implements several security checks to ensure that the memory accesses are correctly authorized and executed. Specifically, SGX records the allocation decisions in the Enclave Page Cache Map (EPCM), which is used to check the EPC access permissions. Also, to protect data integrity, SGX employs the SGX integrity tree (SIT) to store the HMACs of all the data blocks. However, SGX v1 can only provide limited memory for each enclave due to the overhead of SIT. By removing SIT, SGX v2 can support larger memory sizes but with the cost of weakened security protection, especially against hardware replay attacks [8, 10].

Several existing works extend TEEs to disaggregated resources. For example, Composite Enclave [18] enables the enclave to access multiple hardware and I/O devices. DF-TEE [26] extends TEEs to support disaggregated multi-FPGA systems. However, none of the existing solutions support the disaggregated memory architecture.

3 THREAT MODELS

Disaggregated memory systems face several security threats inherited from both the traditional memory systems and the new disaggregated architecture.

- **Privileged software attacks.** On the client side, the malicious software with full control over the software stack can pretend to be the trusted client and access the trusted memory without authorization [8].
- **Data-at-rest attacks.** The attacker with full control over the memory can easily access the memory to modify the in-stored data [9, 13]. Also, the attacker with hardware privilege can move certain data blocks into another position in the memory to raise the splicing or relocation attacks [9].
- **Data-in-transit attacks.** The attacker with no privilege can spoof the data in transmission, record the data previously stored in the memory, or overwrite the memory with the old data (i.e., replay attacks) [8, 9, 19]. In SGX v2, the motherboard manufacturer is considered as a trusted entity, so the replay attack no longer needs to be considered [5]; however, in disaggregated memory systems, since the data is transmitted via untrusted networks, and the replay attacks become significantly more concerning.

In this paper, we do not consider side channel attacks [23] as the original SGX is not designed to defend against them, and the enhancement of side channel resilience on the original SGX can also be applied to *DM-TEE*. Also, this paper does not address denial of service (DoS) attacks and routing attacks, as the mitigation of those attacks belongs to an orthogonal research area.

4 SYSTEM DESIGN

4.1 Design Goals

DM-TEE aims to achieve the following goals in designing a TEE for disaggregated memory systems:

- **G1: Security.** The secured disaggregated memory must keep the same security standards established by SGX [8], i.e., enclave isolation, data confidentiality, and integrity. The system should

consider potential attacks from privileged software and communication channels.

- **G2: Transparency.** The system design should fit into the current SGX architecture, and the enclave can use the disaggregated memory without noticing the implementation details of the disaggregated memory system.
- **G3: Scalability.** The system should be designed without limitations on the number of enclaves and the size of the disaggregated memory for each enclave.
- **G4: Performance.** The system should not incur high performance overhead over the original disaggregated memory system.

4.2 System Components

Figure 2 illustrates the overall *DM-TEE* system architecture, including 3 components: the enclave on the *client* side requests the disaggregated memory; the remote memory manager (i.e., *rMem manager*) in the cloud attests the client and provides the disaggregated memory services; and the *secured memory nodes (SMNs)* are the hardware disaggregated memory entities that connect to the rMem manager.

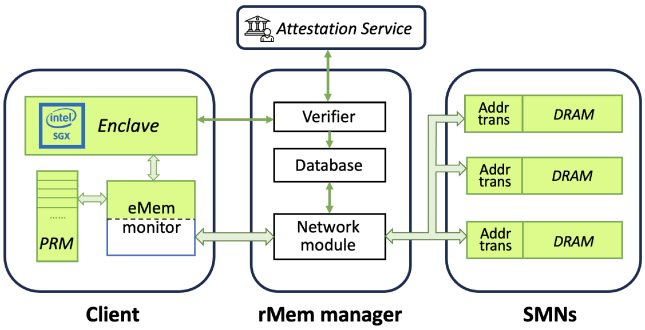


Figure 2: The system overview of *DM-TEE*.

4.2.1 Client. Client is the user entity with the enclave that requests the disaggregated memory. In our design, the disaggregated memory is only for the enclave data. The eMem monitor resides in both the enclave trusted runtime system (tRTS) and the untrusted runtime system (uRTS), and serves as the central component to manage the virtual address space, process the memory requests, and communicate with the rMem manager. With the help of the eMem monitor, the enclave can access the disaggregated memory without knowing the implementation details of the disaggregated memory architecture (G2). The eMem monitor can automatically generate the memory access request, and port the request from the enclave to the rMem manager via ECalls and OCalls. To ensure data integrity, the enclave maintains a customized integrity tree for all the allocated memory, and the data in the disaggregated memory should always be encrypted (G1). Inspired by the PENGLAI enclave [10], we implement a customized mountable Merkle tree (MMT) to maintain the enclave integrity (G1). In MMT, the empty leaf node in the rootTree can be replaced by a subTree when allocating the memory. Also, the subTree for the deallocated memory is substituted with an empty leaf node, effectively reducing the time and the space overhead for integrity check at runtime (G1, G3, G4).

4.2.2 rMem manager. In the cloud service provider, the rMem manager bridges the communication channel between the client and the disaggregated memory. To achieve efficient memory access, we employ a lightweight request authentication mechanism. The rMem manager employs a verifier to attest the enclaves by conducting SGX remote attestation. Similar to JWT [2], the attested enclave will be granted a token signed by the verifier, which can be used for further authentications of memory requests. Each token consists of a payload and a signature. The payload includes the enclave ID and a timestamp. The rMem manager can check the token expiration using the timestamp. The verifier in the rMem manager stores a secret key to sign the token payload and generate the signature, and the key can only be accessed by the rMem manager. The network module within the rMem manager serves as the gateway for client communication, which can support high-speed connections with SMNs via optical fiber (G4). To minimize the database searches and reduce overhead, the network module maintains a routing table to efficiently direct the memory accesses (G4). The routing table records the information of allocated memory, including the enclave IP address, the SMN IP address, and the token information for verification. To protect the memory against physical attacks, the routing table is stored locally and cannot be accessed by third-party entities (G1).

4.2.3 Secured Memory Node (SMN). The SMN consists of DRAM and attached hardware. The attached hardware performs address translation and permission checking. To maintain the security equivalence with the local enclave memory, we implement the EPCM structure [8], which records the page status and the corresponding enclave ID, to verify if the memory page belongs to a specific enclave and the operations are within the appropriate permissions (G1).

4.3 System Workflows

4.3.1 Secured Disaggregated Memory Allocation. Figure 3 illustrates the workflow for the memory allocation in *DM-TEE*. The rMem manager attests the enclave by conducting SGX remote attestation and generates the token for the attested enclave (i.e., the pink arrows in Figure 3). The token contains the payload (i.e., the enclave ID and the timestamp) and the signature, and the rMem manager records the token in its database. When the enclave makes a request for disaggregated memory allocation, the eMem monitor in the tRTS checks the initial virtual address (VA) and determines if the request needs to go to the disaggregated memory. We preserve a VA region in the client specifically for disaggregated memory. The requests for the local memory are managed by the original SGX workflow, as shown in black arrows in Figure 3. The requests for the disaggregated memory are forwarded to the rMem manager via OCalls in the uRTS, which are illustrated as the blue arrows in Figure 3. The uRTS communicates with the rMem manager to send the requests to the remote memory service provider.

To achieve the client memory isolation, the rMem manager verifies the token to determine if the request originates from a trusted client and maintains a routing table to efficiently route the read/write requests to the target SMNs. After verification, the rMem manager searches for the available memory in the SMNs and assigns it to the enclave by: (1) recording the allocation information

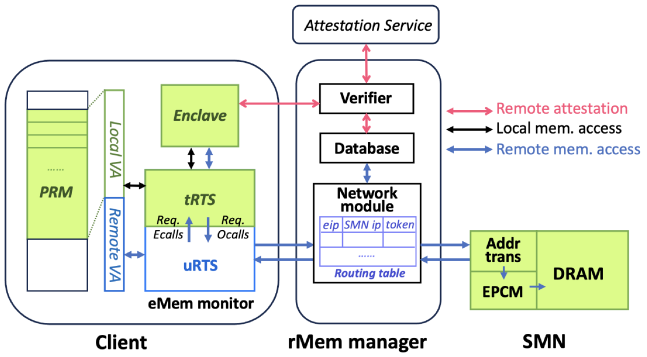


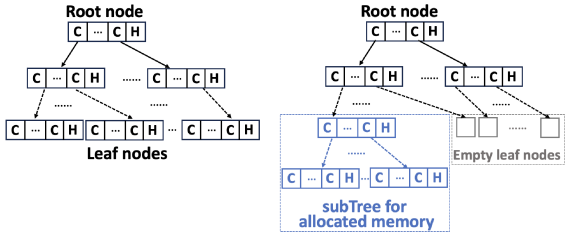
Figure 3: The memory allocation workflow in *DM-TEE*.

in the routing table; (2) notifying the client of the updated VA; and (3) updating the memory status in the database. In the following read/write requests, the client uses the updated VA to access the allocated memory region. The rMem manager also updates the page table and EPCM in the SMN for the ongoing memory access. When the disaggregated memory is allocated, the enclave builds a subTree for the newly allocated memory, replacing an empty leaf node with this subTree in MMT and subsequently updating MMT from leaf to root. To release the allocated memory, the client issues a deallocation request and replaces the subTree with an empty leaf node. Then, the rMem manager updates the database, removes the relevant record from the routing table, and instructs the SMN to update the page table and EPCM accordingly.

4.3.2 Secured Disaggregated Memory Write/Read. To write the data to the allocated disaggregated memory, the eMem monitor first identifies the VA of the writing operation. If the VA falls within the disaggregated memory region, the monitor contacts the rMem manager with the writing request, including the enclave token, the VA, the data size, and the encrypted data. After receiving the request, the rMem manager searches the routing table for the corresponding SMN. In the next step, the SMN translates the VA and executes the EPCM checking. Authorized requests are then permitted to write data to the appropriate memory page. The client is notified once the data writing is complete and updates the MMT accordingly. Similarly, read requests containing valid information will be processed by the corresponding SMN. After the EPCM verification, the SMN sends the data to the client, and the enclave decrypts the data and verifies if the HMACs match with the MMT. In the write/read process, the data remains encrypted, eliminating potential data leakage. EPCM checking ensures memory isolation among different enclaves, and MMT guarantees data integrity.

5 IMPLEMENTATIONS

We implement the eMem monitor on the client side based on the Intel SGX enclave memory manager (EMM) [4]. EMM is running on the SGX v2 platform and supports dynamic enclave memory management with pre-defined APIs. We have extended EMM to accommodate disaggregated memory by implementing two key changes, including VA management and establishing ECalls/OCalls for disaggregated memory requests. Also, the tRTS part in the eMem monitor encapsulates the request with parameters including the enclave ID, the virtual address, the memory size, and the token. The uRTS part works for request transmission only. In the enclave, we deploy an integrity tree in *DM-TEE* based on a customized MMT, as shown in Figure 4, in comparison to the SIT in the original SGX. In SIT, each memory node represents 8 EPCs and stores its HMAC. We leverage such a node structure and integrate it into MMT, and the memory node in MMT can also represent the disaggregated memory pages. In our MMT implementation, the rootTree is assigned a fixed height, and each leaf node can be replaced by a subTree. The subTree is constructed and integrated into the rootTree during the memory allocation, which has the same node structure as the rootTree. During memory deallocation, the subTree is removed from the rootTree and replaced with an empty leaf node.



(a) SIT (b) Customized MMT

Figure 4: The structures of (a) the original SGX integrity tree (SIT) and (b) the customized mountable Merkle tree (MMT) in *DM-TEE* .

We build the SMN based on the fast path in the Clio project [12]. Specifically, we add the additional EPCM checking component after the address translation. The EPCM structure contains the enclave ID, and only the request with the same enclave ID can access the page. As discussed in Section 4.2, the disaggregated memory is designed to store the enclave data only; therefore, all the pages are regular pages, and the page type checking in the original EPCM checking flow [8] is no longer necessary.

6 EXPERIMENTAL RESULTS

6.1 Experimental Setup

We implement *DM-TEE* on a Xilinx ZCU102 FPGA board by referring to the open source Clio implementation [12] (i.e., the fast path design) for the disaggregated memory system. We connect the ZCU102 board with the rMem manager (deployed on an HP Z2 workstation with an Intel Core-i7 CPU and Mellanox ConnectX-5 NIC) using two SFP ports, and each port offers communication with a speed of 10Gbps. Additionally, we adopt an Intel NUC (NUC7CJYH), which supports Intel SGX v2, to play the role of the client issuing disaggregated memory requests. The Intel NUC communicates with the rMem manager in a local area network.

We build an original disaggregated memory system without any security protection as the baseline system, following the system design in Section 2.1. In this system, the client (i.e., Intel NUC) runs the application without SGX and sends requests to the rMem manager

to access disaggregated memory. The rMem manager only transfers requests without authentication. The FPGA board, which manages the disaggregated memory, operates without security checking. Also, we evaluate *DM-TEE* using 4 image processing benchmarks, namely Gaussian blur, Contrast, Gamma, and Compression, with input images of varying sizes ranging from 1 kb to 5 kb.

6.2 Security Analysis

6.2.1 Comparison with Original SGX. Table 1 presents a comparative summary of SGX v1, SGX v2, and *DM-TEE*. In comparison to the original SGX, *DM-TEE* utilizes the disaggregated memory architecture to overcome the enclave memory limitations in SGX v1, effectively offering unlimited memory size. To preserve the isolation, *DM-TEE* utilizes the similar mechanism in SGX (i.e., EPCM verification) to authenticate data ownership and protect the data from unauthorized accesses. Since all memory pages stored remotely must be regular pages, *DM-TEE* eliminated the need of page type checking from the original EPCM verification while maintaining the same security standard. Also, all data stored in the disaggregated memory is encrypted, ensuring confidentiality. Furthermore, *DM-TEE* deploys a customized MMT to protect the data against replay attacks, thereby maintaining the data integrity in the enclave, which is not achieved by SGX v2.

Table 1: Security analysis of SGX v1, SGX v2, and *DM-TEE*.

TEEs	Mem size	Mem isol.	Data conf.	Data inte.
SGX v1	128/256 MB	√	√	√
SGX v2	unlimited	√	√	×
DM-TEE	unlimited	√	√	√

6.2.2 Privileged software attacks. In *DM-TEE*, the rMem manager is the only entity that can communicate with the memory, and the third-party memory access requests cannot reach the memory. Also, the client enclave is authenticated with SGX remote attestation, which prevents the untrusted client from accessing the memory.

6.2.3 Data-at-rest attacks. Similar to the defense against privileged software attacks, only the rMem manager can communicate with the disaggregated memory. Also, the EPCM identifies the ownership of the memory page. The client can only access the pre-allocated memory space, which declines accesses from unauthorized clients and guarantees its isolation. Additionally, the data stored in the disaggregated memory is always encrypted, and its HMAC is recorded in MMT. Therefore, even if an attacker were to physically tamper with the disaggregated memory, it is impossible to access the data in plain text or compromise the data integrity.

6.2.4 Data-in-transit attacks. To defend against the data-in-transit attacks (i.e., replay attacks), *DM-TEE* utilizes the MMT to protect the data integrity. Any data update will trigger an update to the MMT, and the old data cannot pass the verification because its HMAC no longer matches with the current MMT record.

6.3 Performance Evaluation

6.3.1 Memory Allocation/Deallocation Evaluation. We first evaluate the performance of disaggregated memory allocation and deallocation in both our system (i.e., *DM-TEE*) and the original disaggregated memory system (i.e., original). In the evaluation process, the client enclave initiates a memory allocation request, and the rMem manager announces the updated VA to the client. Also, the rMem manager updates the corresponding SMN. We evaluate the allocation and deallocation requests using 1 kb to 8 kb data. As any system calls (e.g., time functions) are prohibited inside the enclave, we create an enclave with an empty ECall that performs no operation. Then, we calculate the time difference between the empty ECall and the memory allocation ECall in both *DM-TEE* and the original disaggregated memory system. The results are presented in Figure 5, which indicate that *DM-TEE* introduces an additional overhead of approximately 21% compared to the original system, which is deemed acceptable given the significantly enhanced security.

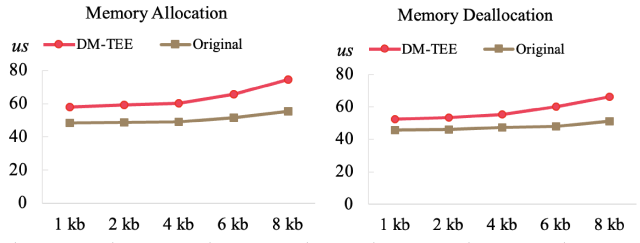


Figure 5: The memory allocation/deallocation overhead.

6.3.2 Memory Read/Write Evaluation. We also evaluate the memory read/write overhead during the execution. We first measure the execution time of an enclave for only memory allocation/deallocation, without any read/write operations. Then, we evaluate the memory read/write operations in the enclave. The time difference between allocation-only and read/write enclaves represents the overhead for the memory read/write, which is shown in Figure 6. *DM-TEE* introduces memory read/write overhead ranging from 15% to 20%, comparing to the original system.

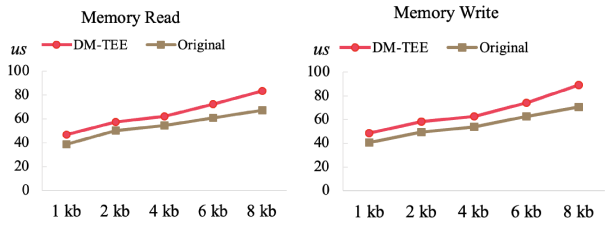


Figure 6: The memory read/write overhead.

6.3.3 Benchmark Evaluation. To evaluate the performance with real workloads, we employ 4 benchmarks, where the input image size scales from 1 kb to 5 kb. In these benchmarks, the disaggregated memory is employed for storing the input images. The evaluation results are shown in Table 2, which indicates that *DM-TEE* introduces

Table 2: Timing evaluation of the benchmarks. The time unit is us.

Benchmarks	1 kb		2 kb		3 kb		4 kb		5 kb	
	Base	DM	Base	DM	Base	DM	Base	DM	Base	DM
Gaussian blur	405.16	446.38	447.62	502.36	486.16	539.21	512.61	570.25	557.12	623.7
Contrast	215.67	246.52	230.20	267.11	250.26	289.74	267.19	305.53	294.66	330.74
Gamma	226.63	252.92	235.21	274.52	253.03	291.33	268.01	308.34	299.12	345.18
Compression	284.67	346.84	327.04	402.62	386.06	469.63	440.17	532.73	497.21	613.38

trivial overhead compared to the original system. Specifically, the Gaussian blur benchmark introduces the least overhead, which is about 11%; the image compression benchmark introduces the most overhead, which is about 21%. In conclusion, our experiment shows that *DM-TEE* introduces acceptable overhead when executing real workloads.

7 CONCLUSION

We have developed *DM-TEE*, a new trusted execution environment to support disaggregated memory systems. *DM-TEE* employs a series of newly designed security components to achieve secure memory allocation and access workflow for disaggregated memory, including an eMem monitor to provide transparent disaggregated memory services to the local enclave, an rMem manager to attest the memory requests and decline untrusted memory accesses, and an EPCM checking process to verify the memory access permission. In addition, we employ a customized MMT structure to ensure the integrity of the allocated data with acceptable complexity. Our evaluations indicate that *DM-TEE* achieves disaggregated memory security with minimal overhead. The project repository of *DM-TEE* is at: https://github.com/hwsel/DM-TEE.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Foundation under award 1912593.

REFERENCES

[1] 2005. ARM Security Technology: Building a Secure System using TrustZone Technology. https://developer.arm.com/documentation/PRD29-GENC-009492/latest/.
[2] 2015. JSON Web Token. https://datatracker.ietf.org/doc/html/rfc7519.
[3] 2022. Enclave Development Overview. https://learn.microsoft.com/en-us/azure/confidential-computing/confidential-computing-enclaves.
[4] 2023. Intel Software Guard Extensions Enclave Memory Manager. https://github.com/intel/sgx-emm/tree/main.
[5] 2023. Runtime Encryption of Memory with Intel Total Memory Encryption–Multi-Key (Intel TME-MK). https://www.intel.com/content/www/us/en/developer/articles/news/runtime-encryption-of-memory-with-intel-tme-mk.html.
[6] Marcos K Aguilera, Kimberly Keeton, Stanko Novakovic, and Sharad Singhal. 2019. Designing far memory data structures: Think outside the box. In *Workshop on Hot Topics in Operating Systems (HotOS)*. 120–126.
[7] Raad Bahmani, Ferdinand Brasser, Ghada Dessouky, Patrick Jauernig, Matthias Klimmek, Ahmad-Reza Sadeghi, and Emmanuel Stapf. 2021. CURE: A security architecture with customizable and resilient enclaves. In *USENIX Security Symposium*. 1073–1090.
[8] Victor Costan and Srinivas Devadas. 2016. Intel SGX explained. *Cryptology ePrint Archive* (2016).
[9] Reouven Elbaz, David Champagne, Catherine Gebotys, Ruby B Lee, Nachiketh Potlapally, and Lionel Torres. 2009. Hardware mechanisms for memory authentication: A survey of existing techniques and engines. *Transactions on Computational Science IV: Special Issue on Security in Computing* (2009), 1–22.
[10] Erhu Feng, Xu Lu, Dong Du, Bicheng Yang, Xueqiang Jiang, Yubin Xia, Binyu Zang, and Haibo Chen. 2021. Scalable memory protection in the PENGLAI enclave. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 275–294.
[11] Juncheng Gu, Youngmoon Lee, Yiwen Zhang, Mosharaf Chowdhury, and Kang G Shin. 2017. Efficient memory disaggregation with infiniswap. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 649–667.
[12] Zhiyuan Guo, Yizhou Shan, Xuhao Luo, Yutong Huang, and Yiying Zhang. 2022. Clio: A hardware-software co-designed disaggregated memory system. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 417–433.
[13] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. 2014. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. *ACM SIGARCH Computer Architecture News* 42, 3 (2014), 361–372.
[14] Youngeun Kwon and Minsoo Rhu. 2019. A disaggregated memory system for deep learning. *IEEE Micro* 39, 5 (2019), 82–90.
[15] Xin Lin, Lingguang Lei, Yuewu Wang, Jiwu Jing, Kun Sun, and Quan Zhou. 2018. A measurement study on linux container security: Attacks and countermeasures. In *Annual Computer Security Applications Conference (ACSAC)*. 418–429.
[16] Fan Mo, Ali Shahin Shamsabadi, Kleomenis Katevas, Soteris Demetriou, Ilias Leontiadis, Andrea Cavallaro, and Hamed Haddadi. 2020. Darknetz: Towards model privacy at the edge using trusted execution environments. In *International Conference on Mobile Systems, Applications, and Services (MobiSys)*. 161–174.
[17] Mendel Rosenblum and Tal Garfinkel. 2005. Virtual machine monitors: Current technology and future trends. *Computer* 38, 5 (2005), 39–47.
[18] Moritz Schneider, Aritra Dhar, Ivan Puddu, Kari Kostiainen, and Srdjan Čapkun. 2022. Composite Enclaves: Towards Disaggregated Trusted Execution. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2022), 630–656.
[19] Dimitrios Skarlatos, Mengjia Yan, Bhargava Gopireddy, Read Sprabery, Josep Torrellas, and Christopher W Fletcher. 2019. Microscope: Enabling microarchitectural replay attacks. In *International Symposium on Computer Architecture (ISCA)*. 318–331.
[20] Chia-Che Tsai, Donald E Porter, and Mona Vij. 2017. Graphene-SGX: A practical library OS for unmodified applications on SGX. In *USENIX Annual Technical Conference (ATC)*. 645–658.
[21] Stavros Volos, Kapil Vaswani, and Rodrigo Bruno. 2018. Graviton: Trusted execution environments on GPUs. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 681–696.
[22] Chenxi Wang, Haoran Ma, Shi Liu, Yuanqi Li, Zhenyuan Ruan, Khanh Nguyen, Michael D Bond, Ravi Netravali, Miryung Kim, and Guoqing Harry Xu. 2020. Semeru: A Memory-Disaggregated managed runtime. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 261–280.
[23] Wenhao Wang, Guoxing Chen, Xiaorui Pan, Yinqian Zhang, XiaoFeng Wang, Vincent Bindschaedler, Haixu Tang, and Carl A Gunter. 2017. Leaky cauldron on the dark land: Understanding memory side-channel hazards in SGX. In *Computer and Communications Security (CCS)*. 2421–2434.
[24] Samuel Weiser and Mario Werner. 2017. SGXIO: Generic trusted I/O path for Intel SGX. In *ACM Conference on Data and Application Security and Privacy (CODASPY)*. 261–268.
[25] Ke Xia, Yukui Luo, Xiaolin Xu, and Sheng Wei. 2021. SGX-FPGA: Trusted execution environment for CPU-FPGA heterogeneous architecture. In *Design Automation Conference (DAC)*. 301–306.
[26] Ke Xia and Sheng Wei. 2023. DF-TEE: Trusted Execution Environment for Disaggregated Multi-FPGA Cloud Systems. In *Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. 1–6.
[27] Jianping Zhu, Rui Hou, XiaoFeng Wang, Wenhao Wang, Jiangfeng Cao, Boyan Zhao, Zhongpu Wang, Yuhui Zhang, Jiameng Ying, Lixin Zhang, et al. 2020. Enabling rack-scale confidential computing using heterogeneous trusted execution environment. In *IEEE Symposium on Security and Privacy (S&P)*. 1450–1465.