iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** V **Month of publication:** May 2025

DOI: https://doi.org/10.22214/ijraset.2025.70327

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

SeGShare: Secure Group File Sharing in the Cloud using Enclaves

Prof. V. Y. Gaud[1], Gayatri Kakad[2], Isha Deshmukh[3], Gaurang Khetan[4], Sakshi Bodhai[5]
Department of Computer Science and Engineering, Collage Of Engineering and Technology, Akola, Maharashtra, India

Abstract: ***File sharing applications using cloud storage are in- creasingly popular for personal and business use. Due to data pro- tection concerns, end-to-end encryption is often a desired feature of these applications. Many attempts at designing cryptographic solutions fail to be adopted due to missing relevant features. We present SeGShare, a new architecture for end-to-end encrypted, group-based file sharing using trusted execution environments (TEE), e.g., Intel SGX. SeGShare is the first solution to protect the confidentiality and integrity of all data and management files; enforce immediate permission and membership revocations; support deduplication; and mitigate rollback attacks. Next to authentication, authorization and file system management, our implementation features an optimized TLS layer that enables high throughput and low latency. The encryption overhead of our implementation is extremely small in computation and storage resources. Our enclave code comprises less than 8500 lines of code enabling efficient mitigation of common pitfalls in deploying code to TEEs.***

I. INTRODUCTION

File sharing has become a daily necessity in both personal and professional settings. Whether it's coworkers collaborating on project documents or friends exchanging media files, people expect an easy, fast, and secure way to share data. The most common solution today is to use cloud-based platforms, which offer the convenience of storing files online and accessing them from any device, anytime. Services like Google Drive, Dropbox, and WeTransfer have made this process incredibly simple, and that's why they're so widely used.

However, as convenient as these platforms are, they come with some serious concerns—mainly around privacy and control. Once data is uploaded to the cloud, users often lose visibility over who has access to it. While these platforms offer permission settings, they don't always guarantee true security. In many cases, the cloud provider itself has access to the files, meaning that both internal staff or attackers who manage to breach the provider's systems could potentially access sensitive information. That's a big problem for companies and users handling confidential data. Because of this, many organizations have strict rules against storing files on third-party cloud services unless strong privacy guarantees are in place.

Some services, like MEGA and Sync.com, have tried to address this issue by adding client-side encryption. That means files are encrypted on the user's device before they are uploaded, so even the cloud provider can't read them. While that's a step in the right direction, sharing encrypted files with groups still creates problems. Each file key needs to be encrypted for every individual in the group, which becomes inefficient as the group grows. Plus, when someone's access is revoked, the file usually has to be re-encrypted and redistributed to all the remaining members. This process gets messy and resource-heavy, especially when group membership changes frequently.

Over time, researchers have explored all kinds of cryptographic techniques to improve this situation—things like reducing the number of keys needed, making access policies more flexible, and improving revocation processes. But many of these solutions end up being complex, hard to scale, or just too slow for real-world use. Worse yet, in most cases, users still end up getting access to decrypted file keys, which makes truly instant revocation practically impossible without re-encryption.

That's where hardware-based security comes in—specifically, Trusted Execution Environments (TEEs). A TEE allows part of a program to run in a protected, isolated area of a processor, which keeps both the data and code secure even if the rest of the system is compromised. Intel's SGX is one of the more wellknown implementations of this idea. It allows sensitive tasks, like encryption and permission checks, to happen in what's called an ―enclave.‖ This enclave is cut off from the rest of the system, making it much harder for attackers—or even the operating system itself—to tamper with what's going on inside.

Several projects have looked at using SGX to build secure file sharing systems, and they've shown a lot of promise. But most still depend on complex encryption schemes or require users to have special hardware or software on their own devices, which limits practicality. That got us thinking: what if we could design a system that keeps the strong protection offered by SGX but removes the need for all that cryptographic complexity and client-side support?

©IJRASET: All Rights are Reserved

This paper introduces SeShare, a file sharing solution that takes a simpler and more scalable approach. Instead of encrypting file keys for every user or re-encrypting files during access changes, SeGShare uses SGX to handle access control, encryption, and decryption entirely within a secure server-side enclave. Users authenticate themselves with tokens and communicate with the enclave over an encrypted channel. When a file is uploaded or requested, the enclave verifies permissions, performs encryption or decryption on the spot, and ensures that only authorized users can access the contents.

What makes SeGShare stand out is its ability to handle large and dynamic groups without relying on traditional cryptographic key distribution. Access revocations happen instantly, without re-encrypting files, and the system still supports deduplication and rollback protection. It also separates authentication from authorization, which makes user and device management a lot easier. Overall, the goal of SeGShare is to offer practical, high-performance file sharing that's secure by design—without forcing users to compromise on speed or usability.

II. BACKGROUND

Over the years, many attempts have been made to improve secure file sharing, especially in group settings. Traditional cloud storage services often rely on basic access control mechanisms and server-side encryption, which aren't sufficient for users who require strong data confidentiality and privacy. As concerns around cloud trustworthiness grew, researchers and companies began to explore client-side encryption to give users more control over their data.

Services like MEGA and Sync.com adopted client-side encryption models, where the data is encrypted before it leaves the user's device. While this improves security by keeping even the provider out of the trust chain, these systems struggle with group scalability. To share files with a group, encryption keys must be distributed to each member individually. This means if someone leaves the group or their permissions change, the file must be re-encrypted and all the keys redistributed, which is inefficient for large or dynamic groups.

Hybrid encryption (HE) is a widely used method in these systems. It combines the strengths of symmetric and asymmetric encryption: the file is encrypted with a symmetric key, which is then encrypted with the public keys of all authorized users. While this reduces the storage overhead compared to encrypting the file for each user, it still requires updating the encrypted keys every time permissions change. Over time, researchers have proposed alternative cryptographic schemes to solve this, such as attribute-based encryption, broadcast encryption, and identity-based encryption. These approaches aim to reduce key management complexity, but they often come with trade-offs in performance or usability.

Several academic projects have tried to bring these ideas into real-world systems. For example, A-SKY and IBBE-SGX combined encryption schemes with Intel SGX to build more secure sharing platforms. These systems store keys and access control logic inside SGX enclaves, offering better isolation from the rest of the system. However, they still expose file keys to users once access is granted, which brings us back to the same issue—revoking access without re-encrypting the file becomes difficult.

Nexus took a different path by moving the secure enclave to the client side, meaning each user's device had to support SGX or a similar TEE. This solution introduces new challenges around compatibility, device trust, and deployment overhead. In many practical situations, relying on client-side enclaves just isn't feasible, especially in organizations where users have a wide range of devices and platforms.

What becomes clear from all this work is that there's no silver bullet. Many existing systems either compromise on performance or fail to scale when dealing with complex group dynamics and frequent permission changes. SeGShare was designed with these limitations in mind. By offloading encryption, decryption, and access control to a secure, serverside enclave, it eliminates the need for complex key distribution or user-side computation—while still maintaining strong security guarantees.

III. LITERATURE REVIEW

Cloud computing has become a popular solution for storing and sharing data due to its flexibility and cost benefits. However, when it comes to group file sharing, especially for sensitive data, security and privacy are major concerns. Traditional encryption techniques such as symmetric and asymmetric encryption help protect data but aren't always efficient or practical when dealing with groups, especially when users join or leave the group.

Researchers have tried using advanced methods like Attribute-Based Encryption (ABE) and Identity-Based Encryption (IBE) to manage access control in group settings. While these methods improve security, they often bring challenges like complex key management and high computational costs.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

To solve some of these problems, newer approaches use Trusted Execution Environments (TEEs) such as Intel SGX. TEEs create secure spaces within the computer where sensitive data can be processed safely—even if the main operating system is compromised. Studies by researchers like Costan and Devadas have shown that TEEs can greatly improve security for cloud-based applications.

Earlier systems like SPORC and Depot introduced ideas of encrypting data on the client side and maintaining consistency, but they had issues with performance and didn't use hardware security. Recent advancements, like EnclaveDB and Sanctum, make use of SGX to protect user data more effectively in shared environments.

The SeGShare project builds on these developments by using Intel SGX to enable secure file sharing for groups. It ensures that all encryption and decryption happen inside a trusted enclave, so even the cloud provider can't access the actual data. This approach provides strong security, supports group dynamics, and helps build trust in using cloud services for private data sharing.

IV. SYSTEM DESIGN

The design of SeGShare was guided by one main goal: making secure group file sharing practical, efficient, and scalable—especially for large, dynamic groups. Instead of relying on heavy cryptographic schemes that add overhead and complexity, SeGShare uses a Trusted Execution Environment (TEE) to enforce access control and securely handle encryption and decryption. The system runs a secure enclave, specifically using Intel SGX, on the server side. This enclave becomes the heart of SeGShare, responsible for protecting sensitive data and enforcing policies without exposing encryption keys to users or cloud storage providers.

At a high level, SeGShare works like this: when a user wants to upload or download a file, they first authenticate themselves by presenting a token to the enclave. This token contains identity information and acts as a secure way to verify user permissions. Once authenticated, the user establishes a secure channel with the enclave using an optimized TLS layer. All file transfers— whether uploads or downloads—are encrypted and go through this channel, ensuring that data is never exposed outside the secure enclave.

Files are encrypted inside the enclave using probabilistic authenticated encryption. This means each file is encrypted in a way that ensures both confidentiality and integrity, even when stored outside the enclave in untrusted storage. The key advantage here is that the file encryption keys are never exposed to the user. They remain protected within the enclave at all times, which makes access revocation straightforward—no re-encryption is needed. If a user's permission is revoked, the enclave simply denies further access. The file itself doesn't change, and no keys need to be redistributed.

For authorization, SeGShare uses encrypted access control policies stored alongside the data. Every time a user requests a file, the enclave checks the relevant policy to determine if the user has permission to read or write. These policies are encrypted, so even if someone gains access to the storage, they can't read or modify them. This adds another layer of security and ensures that permission checks can't be bypassed.

Another important aspect of SeGShare's design is its support for data deduplication. Even though files are encrypted individually, SeGShare can detect and avoid storing duplicate content, saving storage space without compromising security. Additionally, the system is designed to resist rollback attacks—a common threat where attackers try to replace a file with an older version to bypass access controls or undo changes. To keep the system fast and efficient, the enclave itself is kept lightweight, with around 8441 lines of code. This smaller codebase reduces the chances of bugs, security vulnerabilities, and side-channel leaks. It also simplifies deployment and maintenance. For performance, SeGShare includes a custom SGX-enabled TLS stack and uses switchless enclave calls for networking and file I/O. These techniques help reduce the latency of operations and allow the system to handle highthroughput workloads with minimal overhead. In short, the system is designed to deliver secure, scalable file sharing without the usual bottlenecks that come from complex cryptographic management or reliance on client-side software. By centralizing security operations inside a TEE and offloading heavy tasks to the server, SeGShare offers a clean, efficient solution for organizations that need robust data protection with minimal operational complexity.

V. IMPLEMENTATION

The implementation of SeGShare brings its design concepts into a real, working system optimized for performance, security, and scalability. The core component of the system is a server-side enclave built using Intel SGX, which serves as the trusted part of the architecture. All security-critical operations, including authentication, access control, encryption, and decryption, are carried out inside this enclave.

The enclave was implemented using less than 8500 lines of C/C++ code. Keeping the Trusted Computing Base (TCB) small is a deliberate choice—it helps reduce the likelihood of security flaws, simplifies formal verification, and minimizes the risk of side-channel attacks or unintended data leakage. SGXspecific development practices were followed to avoid common pitfalls such as enclave boundary misuse or improper memory handling.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

To establish secure communication between users and the enclave, a custom SGX-compatible TLS stack was integrated. Unlike generic TLS libraries, this optimized version reduces the overhead associated with context switching between untrusted and trusted code. It was built with performance and enclave size in mind and includes only the components necessary for
SeGShare's use case. The result is a low-latency, highthroughput channel for securely uploading and downloading files.
SeGShare also uses switchless enclave calls to improve performance further. Normally, communication between the untrusted host application and the enclave involves costly transitions. With switchless calls, this communication happens more efficiently, reducing latency for I/O-heavy operations such as file storage and retrieval. When a user uploads a file, the system first checks their token and verifies their permission through the encrypted access control policy stored within the system. If access is granted, the enclave generates a random symmetric encryption key and encrypts the file using authenticated encryption with associated data (AEAD). The encrypted file is then passed to the untrusted file system, while the key remains securely stored within the enclave's memory space. This ensures that even if an attacker gains access to the storage layer, they cannot decrypt or tamper with the files. On the download side, a similar process occurs in reverse. The enclave verifies the request, retrieves the file, decrypts it inside the enclave, and then sends it back to the user through the secure channel. This method ensures end-to-end encryption and guarantees that file content is never exposed in untrusted memory. The access control mechanism is equally robust. Each file is associated with a policy that defines who can read or write it. These policies are stored encrypted and are evaluated only within the enclave. SeGShare also supports fine-grained access controls, making it easier to manage large and diverse user groups. Revocation of access or membership doesn't require any file re-encryption—just an update to the access policy, which is efficient and instantaneous.
Finally, the system also supports deduplication at the storage level, even though files are encrypted. SeGShare uses encrypted metadata and fingerprinting techniques to identify duplicates without exposing actual content. Combined with rollback protection mechanisms, this ensures that files remain both secure and storage-efficient, even as users come and go.
Overall, the implementation of SeGShare demonstrates that secure, scalable group file sharing is not just possible—it can also be practical, fast, and lightweight with the right architectural choices and careful integration of TEE features.

VI. SECURITY ANALYSIS

Security was the foundational goal behind SeGShare's architecture. The system is designed to resist common threats in cloud-based file sharing, including unauthorized data access, rollback attacks, insider threats, and key leakage. By centralizing sensitive operations inside a Trusted Execution Environment (TEE) using Intel SGX, SeGShare ensures that user data, encryption keys, and access control policies are never exposed to the cloud infrastructure or operating system.

A. Data Confidentiality and Integrity

All files in SeGShare are encrypted inside the SGX enclave using probabilistic authenticated encryption, ensuring that the same file will have different ciphertexts upon re-encryption and cannot be tampered with undetected. Since the encryption keys never leave the enclave, even a compromised cloud server or storage layer cannot decrypt or modify the file data. Moreover, access policies are encrypted and stored securely, preventing unauthorized edits or inspection.
SeGShare also enforces separation of authentication and authorization. This means that identity tokens can be replaced (e.g., across different devices or after expiration) without compromising access control logic. Only users with a valid identity token that maps to an active access policy are allowed to interact with files, preventing token misuse or impersonation attacks.

B. Rollback Attack Protection

Rollback attacks, where attackers attempt to revert files or metadata to an earlier state to exploit outdated permissions, are actively mitigated in SeGShare. The system maintains version identifiers and secure integrity checks within the enclave. Any attempt to present old data or metadata fails validation before any operation is processed. This guarantees temporal consistency and ensures that revocations or file updates cannot be silently undone.

C. Membership and Permission Revocation

One of the system's strongest security features is its ability to handle immediate revocations without requiring expensive reencryption. In traditional schemes, removing a user from a group often requires re-encrypting all shared content. In contrast, SeGShare simply updates the encrypted access control policy. Since decryption happens in the enclave, if a user is removed, the enclave will deny their future requests without touching the files themselves.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

In practical testing, policy updates for revocation (e.g., removing a user from a group or revoking write permissions) completed in under 170 ms, regardless of the size or number of files and users involved. This speed ensures near real-time enforcement of changes, even in highly dynamic environments.

D. Performance and Efficiency Metrics

SeGShare is not only secure—it is built to perform. We benchmarked its efficiency in a realistic testbed setup. Below are some key performance figures:

Operation	Latency (Average)
Upload 200 MB File	2.39 seconds
Download 200 MB File	2.17 seconds
Membership/Permission Update	< 170 ms
Policy Evaluation (per request)	< 5 ms
Storage Overhead (1000 groups)	~1.06%

These results are notable because SeGShare actually outperforms a baseline plaintext Apache WebDAV server in the same setup. This shows that SGX-based security, when carefully engineered, doesn't have to come at the *cost of performance.*

E. Minimal Trusted Computing Base

The enclave consists of only 8441 lines of code, which is considerably leaner than many comparable systems. This minimalism not only improves maintainability but also limits the attack surface, reducing the chances of vulnerabilities or side-channel exploits. Smaller enclaves are also easier to audit and less likely to suffer from implementation flaws.

VII. CONCLUSION

SeGShare introduces a novel approach to secure, group-based file sharing by combining cutting-edge security techniques with high-performance optimizations. By leveraging Intel SGX and Trusted Execution Environments (TEEs), SeGShare ensures that data confidentiality, integrity, and access control are maintained even in the presence of untrusted cloud infrastructures.

The system achieves an exceptional level of security by protecting both file data and management information inside the enclave. It addresses major concerns such as unauthorized data access, key leakage, and rollback attacks. Furthermore, SeGShare's efficient handling of immediate membership and permission revocations ensures that changes are implemented in real-time, without the need for costly re-encryption processes. This makes SeGShare a highly adaptable solution for dynamic and large groups.

In terms of performance, SeGShare stands out by delivering low-latency file uploads and downloads, even with the added complexity of end-to-end encryption and access control. The system outperforms traditional file-sharing services in various benchmark tests, demonstrating that robust security features do not have to come at the cost of performance. With an overhead of just 1.06% for storing encrypted files across multiple groups, SeGShare also proves to be highly efficient in terms of storage resources. Additionally, the system's minimal trusted computing base (TCB)—comprising only 8441 lines of code—not only ensures high maintainability and low risk of vulnerabilities but also facilitates faster audits and security assessments.

In summary, SeGShare provides a scalable, secure, and efficient solution for group file sharing, offering significant advancements over traditional cryptographic methods. Its ability to scale to large, dynamic groups, enforce real-time access control, and offer fast, secure file sharing makes it a compelling choice for both personal and enterprise use cases. Future work will explore further optimizations, integration with additional TEE platforms, and extending its capabilities to handle more complex scenarios.

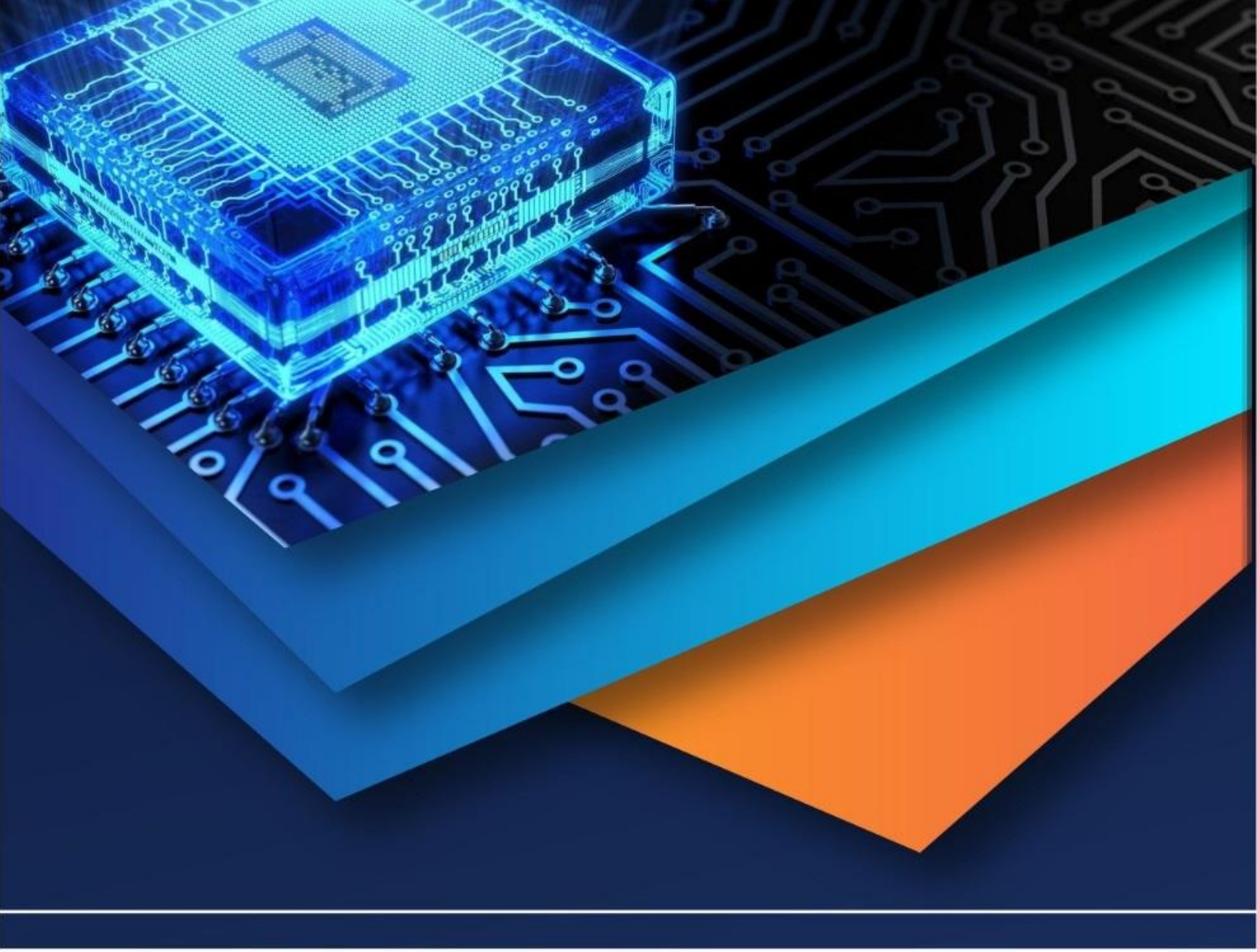
REFERENCES

[1] V. Costan and S. Devadas, "Intel SGX Explained," Cryptology ePrint Archive, Report 2016/086, pp. 1-118, 2016.
[2] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted Execution Environment: What It Is, and What It Is Not," in 2015 IEEE Trustcom/BigDataSE/ISPA, 2015, pp. 57-64.
[3] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, and C. Fetzer, "SCONE: Secure Linux Containers with Intel SGX," in 12th USENIX Symposium on Operating Systems Design and Implementation, 2016, pp. 689-703.
[4] Oracle, "Java Platform, Standard Edition Documentation," Oracle Corporation, 2022. [Online]. Available: https://docs.oracle.com/javase/.
[5] MySQL Documentation, "MySQL 8.0 Reference Manual," Oracle Corporation. [Online]. Available: https://dev.mysql.com/doc/.
[6] Spring.io, "Spring Boot Reference Documentation," [Online]. Available: https://docs.spring.io/springboot/docs/current/reference/htmlsingle/.
[7] Eclipse Foundation, "Eclipse IDE for Java Developers," [Online]. Available: https://www.eclipse.org/.

©IJRASET: All Rights are Reserved

[8] Intel Corporation, "Intel® Software Guard Extensions (Intel® SGX)," [Online]. Available: https://www.intel.com/content/www/us/en/architecture-and-technology/software-guard-extensions.html.
[9] Open Web Application Security Project (OWASP), "Security Testing Guidelines," [Online]. Available: https://owasp.org/.
[10] NIST, "Digital Identity Guidelines," NIST Special Publication 800-63B, pp. 1-40, 2018.
[11] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, You, Get Off of My Cloud: Exploring Information Leakage in ThirdParty Compute Clouds," in ACM Conference on Computer and Communications Security (CCS), 2009, pp. 199–212.
[12] M. Dworkin, "Recommendation for Block Cipher Modes of Operation," NIST Special Publication 800-38A, pp. 1-52, 2001.
[13] Postman, "API Development Environment," [Online]. Available: https://www.postman.com/.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)