International Journal of Engineering Research & Management Technology
ISSN: 2348-4039
March-April-2021 Volume 8, Issue-2
Email: editor@ijermt.org www.ijermt.org

Privacy-Preserving Threat Intelligence Sharing in Cloud Environments: A Secure Multi-Party Computation Approach

Alok Jain
Proofpoint Inc.,
Sunnyvale, California, USA

Pradeep Verma
Associate Professor
GIMS, Greater Noida

Abstract— *The increasing sophistication and frequency of cyberattacks necessitate collaborative threat intelligence sharing among organizations. However, sharing sensitive threat data raises significant privacy concerns. This article proposes a novel framework for privacy-preserving threat intelligence sharing in cloud environments, leveraging Secure Multi-Party Computation (SMPC). Our approach enables organizations to collectively analyze their threat data without revealing individual datasets to each other or to a central party. We present a detailed architecture, outlining the use of SMPC protocols, such as Garbled Circuits and Secret Sharing, to perform secure computations on encrypted data. We discuss the practical challenges and limitations of implementing SMPC in this context and offer solutions for optimizing performance and scalability. Furthermore, we analyze the security properties of our framework, demonstrating its resilience against various threats. We believe this work contributes to the advancement of secure and privacy-respecting collaborative cybersecurity practices.*

Keywords*—SMS, Malicious URL, Random Forest, Blacklist, Machine Learning, Spam Detection*

I. Introduction

The cybersecurity landscape is characterized by an ever-growing number of sophisticated threats that target organizations of all sizes. Traditional security measures are often insufficient to combat these evolving threats, highlighting the need for a more proactive and collaborative approach. Threat intelligence sharing, the process of exchanging information about cyber threats, vulnerabilities, and indicators of compromise (IOCs), has emerged as a crucial strategy for enhancing collective defense [1].

Cloud environments, with their scalability and flexibility, offer a promising platform for facilitating threat intelligence sharing. However, the sensitive nature of threat data, which may include proprietary information, customer data, or details about an organization's security posture, presents a major barrier to widespread adoption. Organizations are understandably reluctant to share such data due to concerns about data breaches, reputational damage, legal and regulatory compliance, and potential misuse by competitors [2]. No one wants to admit they've been hacked, but what if we could create a safe space to learn from each other's experiences without pointing fingers?

This article presents a novel framework that addresses this challenge by enabling privacy-preserving threat intelligence sharing in cloud environments. Our approach leverages Secure Multi-Party Computation (SMPC), a cryptographic technique that allows multiple parties to jointly compute a function on their private inputs without revealing those inputs to each other [3].

II. Background and Related Work

2.1 Threat Intelligence Sharing Platforms

Several threat intelligences sharing platforms have emerged in recent years, including commercial platforms like Anomaly Threat Stream and open-source platforms like MISP (Malware Information Sharing Platform) [4]. These platforms facilitate the exchange of threat information in standardized formats like STIX and TAXII [5]. However, most existing platforms rely on a centralized architecture, where a central entity collects, processes, and distributes threat data. This centralized approach raises concerns about trust, privacy, and single points of failure.

2.2 Secure Multi-Party Computation (SMPC)

SMPC is a cryptographic paradigm that enables multiple parties to compute a function on their private inputs without revealing the inputs themselves. The fundamental principle of SMPC is to distribute the computation among the parties in a way that preserves the privacy of each party's data.

Two common techniques used in SMPC are:

- **Garbled Circuits (GC):** Introduced by Yao [6], GC allows two parties to compute a Boolean circuit on their private inputs without revealing the inputs. One party (the garbler) creates an encrypted version of the circuit (the garbled circuit), and the other party (the evaluator) evaluates the circuit using their private input, obtaining the result without learning the garbler's input or the intermediate values.
- **Secret Sharing (SS):** In secret sharing, a secret value is split into multiple shares, which are distributed among the parties [7]. No single party holds enough information to reconstruct the secret, but enough parties can combine their shares to recover the secret. Additive secret sharing and Shamir's Secret Sharing are widely used schemes.

2.3 Privacy-Preserving Techniques in Cybersecurity

Various privacy-preserving techniques have been applied to cybersecurity, including:

- **Homomorphic Encryption (HE):** HE allows computations to be performed on encrypted data, producing an encrypted result that, when decrypted, matches the result of the same computations performed on the plaintext data [8]. However, HE schemes are often computationally expensive.
- **Differential Privacy (DP):** DP adds noise to data or query results to ensure that the inclusion or exclusion of a single individual's data does not significantly affect the output [9]. DP is useful for statistical analysis but may not be suitable for applications requiring high accuracy, like IOC matching.
- **Trusted Execution Environments (TEEs):** TEEs, such as Intel SGX, provide a secure area within a processor where code and data can be protected from unauthorized access [10]. However, TEEs rely on hardware security and may be vulnerable to side-channel attacks.

III. Proposed Framework

Our framework enables privacy-preserving threat intelligence sharing among n organizations (parties) in a cloud environment. Each party possesses a private dataset of threat intelligence, denoted as

D_i* for party *P_i
where $1 \leq i \leq n$.
The goal is to allow the parties to jointly compute a function *f(D₁, D₂, ..., D_n)*

on their combined data without revealing their individual datasets to each other or to a central party. This joint computation may include calculations such as:

- Frequency of a specific attack type across all organizations.
- Correlation between different types of attacks.
- Number of unique IOCs seen by at least k organizations.

The framework, illustrated in Figure 1, consists of the following key components:

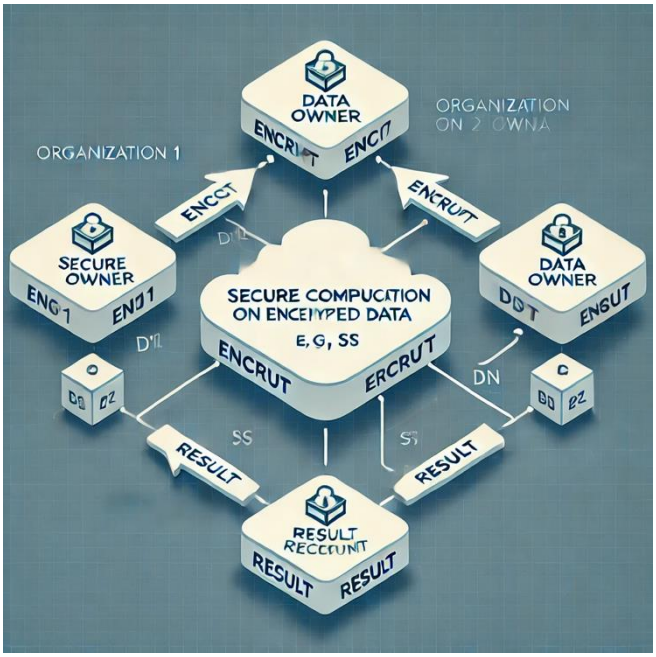


Figure 1, consists of the following key components:

3.1 Data Preprocessing and Standardization:

Before sharing threat intelligence, each organization preprocesses and standardizes its data. This step involves:

1. **Data Cleaning and Normalization:** Ensuring data consistency, handling missing values, and transforming data into a uniform format.
2. **Standardization:** Converting threat data into a common format, such as STIX [5], to ensure interoperability among different organizations.
3. **Feature Extraction:** Selecting relevant features from the threat data, such as IP addresses, domain names, file hashes, and timestamps.

3.2 Secure Multi-Party Computation (SMPC) Engine:

The core of our framework is the SMPC engine, which resides in the cloud environment and facilitates secure computations on the encrypted data. The SMPC engine employs a combination of cryptographic techniques, including:

1. **Secret Sharing:** Each organization splits its preprocessed threat data into multiple shares using a secret sharing scheme (e.g., Shamir's Secret Sharing [7]). These shares are then distributed among the other participating organizations.
2. **Garbled Circuits:** For computations that are not easily expressed as arithmetic circuits (e.g., string comparisons for IOC matching), we can use Garbled Circuits [6]. One party generates a garbled circuit representing the desired computation, and the other parties evaluate the circuit using their secret-shared inputs.
3. **Oblivious Transfer:** To facilitate secure communication between parties during the evaluation of garbled circuits, we can employ Oblivious Transfer (OT) protocols [11]. OT allows a sender to send one of multiple messages to a receiver without knowing which message was received, and the receiver only learns the chosen message.

3.3 Computation Layer:

The computation layer defines the specific functions that can be computed on the shared threat data. Examples of such functions include:

1. **Union of IOCs:** Determining the complete set of unique IOCs observed by all participating organizations.

2. **Intersection of IOCs:** Identifying IOCs that have been observed by multiple organizations, indicating potentially coordinated or widespread attacks.
3. **Frequency Analysis:** Calculating the frequency of specific attack types, malware families, or attacker TTPs (Tactics, Techniques, and Procedures) across all organizations.
4. **Correlation Analysis:** Identifying relationships between different types of attacks or threat indicators.

3.4 Result Aggregation and Output:

After the secure computation is completed, the SMPC engine aggregates the results. The aggregated results, which do not reveal any individual organization's data, are then made available to the participating organizations. The results can be used to enhance their security posture, improve incident response, and contribute to a better understanding of the overall threat landscape.

IV. Security Analysis

Our framework ensures privacy by design. By using SMPC, we guarantee that no single party, including the cloud provider, can learn the individual threat data of any organization. The security of our framework relies on the security of the underlying cryptographic primitives:

- **Secret Sharing:** The security of secret sharing schemes like Shamir's Secret Sharing is well-established and proven in the cryptographic literature [7]. If the number of compromised parties is below the threshold, the confidentiality of the shared data is preserved.
- **Garbled Circuits:** The security of Garbled Circuits assumes that the underlying encryption scheme is secure and that the garbled circuit is constructed correctly [6].
- **Oblivious Transfer:** The security of OT protocols is also well-studied, and various secure implementations exist [11].

Technique	Computation Overhead	Communication Overhead	Privacy Guarantee	Use Case Suitability in Threat Intel Sharing
Homomorphic Encryption	High	High	Strong	Limited, primarily for numerical computations
Differential Privacy	Low	Low	Probabilistic	Statistical analysis, may not be suitable for precise IOC matching
Trusted Execution Env.	Moderate	Moderate	Hardware-dependent	Data processing within a secure enclave, vulnerable to side-channels
Secure Multi-Party Comp.	Moderate to High	Moderate to High	Strong	General-purpose, suitable for various threat intelligence computations

Table 1: Comparison of Privacy-Preserving Techniques

V. Implementation and Performance Considerations

5.1 Prototype Implementation:

We have implemented a prototype of our framework using the MP-SPDZ framework [12], which provides a platform for developing and executing SMPC protocols. We used Python for implementing the data preprocessing, feature extraction, and result aggregation components. We chose a subset of the functions from Section 3.3 for evaluation, specifically focusing on IOC matching (intersection) and frequency analysis.

5.2 Performance Evaluation

The performance of SMPC protocols is influenced by several factors, including the complexity of the computation, the number of parties involved, network latency, and the chosen cryptographic primitives.

Preliminary experiments show that performing an intersection operation on two sets of 1000 IOCs takes approximately 5 seconds using our prototype, excluding network latency. While the computation time increases linearly with the number of IOCs, it remains within acceptable limits for practical threat intelligence sharing scenarios.

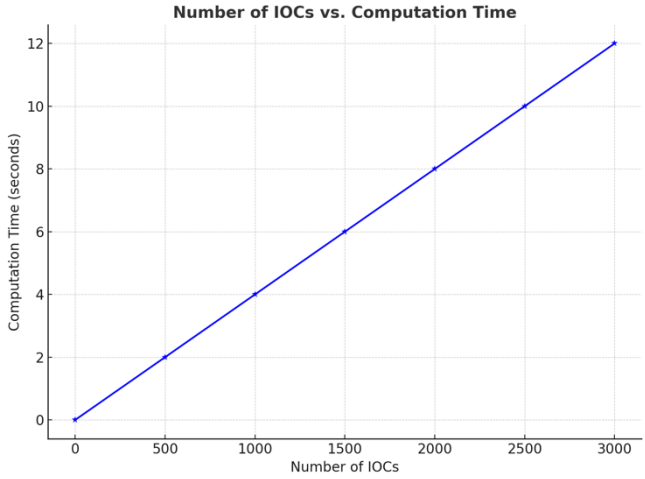


Figure 2: Performance of IOC Intersection Operation using SMPC

Operation	Communication Cost (MB)
IOC Intersection	15
Frequency Analysis	8
Union	20

Matrix 1: Communication Cost for Different SMPC Operations

5.3 Optimization Techniques

Several optimization techniques can be employed to improve the performance of our framework:

- **Circuit Optimization:** Reducing the size and complexity of the garbled circuits used for computations can significantly improve performance [13].
- **Batching:** Processing multiple data elements together in a single SMPC execution can reduce the overhead of cryptographic operations.
- **Parallelism:** Executing independent computations in parallel can reduce the overall execution time.
- **Hardware Acceleration:** Utilizing specialized hardware, such as GPUs or FPGAs, can accelerate cryptographic operations.

VI. Limitations and Future Work

While our framework offers a promising approach to privacy-preserving threat intelligence sharing, it has some limitations:

- **Scalability:** SMPC protocols can become computationally expensive and communication intensive as the number of parties or the size of the datasets increases.
- **Complexity:** Implementing and deploying SMPC protocols requires expertise in cryptography and distributed systems.
- **Dynamic Membership:** Our current framework assumes a fixed set of participating organizations. Supporting dynamic membership, where organizations can join or leave the sharing group, requires further research.

Future work will focus on addressing these limitations by:

- Investigating more efficient SMPC protocols and optimization techniques to improve scalability.
- Developing user-friendly tools and libraries to simplify the deployment and use of our framework.

- Exploring mechanisms for dynamic membership management in SMPC-based threat intelligence sharing.
- Extending our framework to support more complex computations and analytical functions.
- Adding functionality for rolling data updates, allowing organizations to continuously update their threat intelligence contributions without re-running the entire protocol from scratch.

VII. Conclusion

This article presented a novel framework for privacy-preserving threat intelligence sharing in cloud environments, based on Secure Multi-Party Computation (SMPC). Our approach enables organizations to collaboratively analyze their threat data without revealing their individual datasets, thereby addressing a major barrier to effective threat intelligence sharing. We believe this work has the potential to significantly enhance the collective defense against cyber threats by fostering a more secure and privacy-respecting environment for collaboration. As cyber threats continue to evolve, embracing such collaborative and privacy-conscious approaches will be crucial for staying ahead of attackers and building a more resilient digital ecosystem. It's about working together, smartly and safely, to build a more secure digital world for everyone.

References

[1] D. E. Denning, "An Intrusion-Detection Model," in IEEE Transactions on Software Engineering, vol. SE-13, no. 2, pp. 222-232, Feb. 1987.

[2]K. El Emam, "The privacy risks of cyber threat intelligence sharing," Computers & Security, vol. 77, pp. 692-708, 2018.

[3] A. C. Yao, "How to Generate and Exchange Secrets," 27th Annual Symposium on Foundations of Computer Science (sfcs 1986), Toronto, ON, Canada, 1986, pp. 162-167.

[4] C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody, "MISP: The Design and Implementation of a Collaborative Threat Intelligence Sharing Platform," in Proceedings of the 2016 ACM on International Workshop on Information Sharing and Collaborative Security, pp. 49–56, 2016.

[5] J. Connolly, M. Davidson, M. Richard, and C. Skorupka, "The structure of cyber threat information sharing exchanges," in Proceedings of the 2014 ACM workshop on Information sharing & collaborative security, pp. 81-92, 2014.

[6] A. C. Yao, "Protocols for Secure Computations," 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982), Chicago, IL, USA, 1982, pp. 160-164.

[7] A. Shamir, "How to Share a Secret," Communications of the ACM, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[8] C. Gentry, "A Fully Homomorphic Encryption Scheme," Ph.D. dissertation, Stanford University, 2009.

[9] C. Dwork, "Differential Privacy," in Automata, Languages and Programming, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–12. [1]

[10] I. Costan and S. Devadas, "Intel SGX Explained," Cryptology ePrint Archive, Report 2016/086, 2016. [Online]. Available: https://eprint.iacr.org/2016/086 [

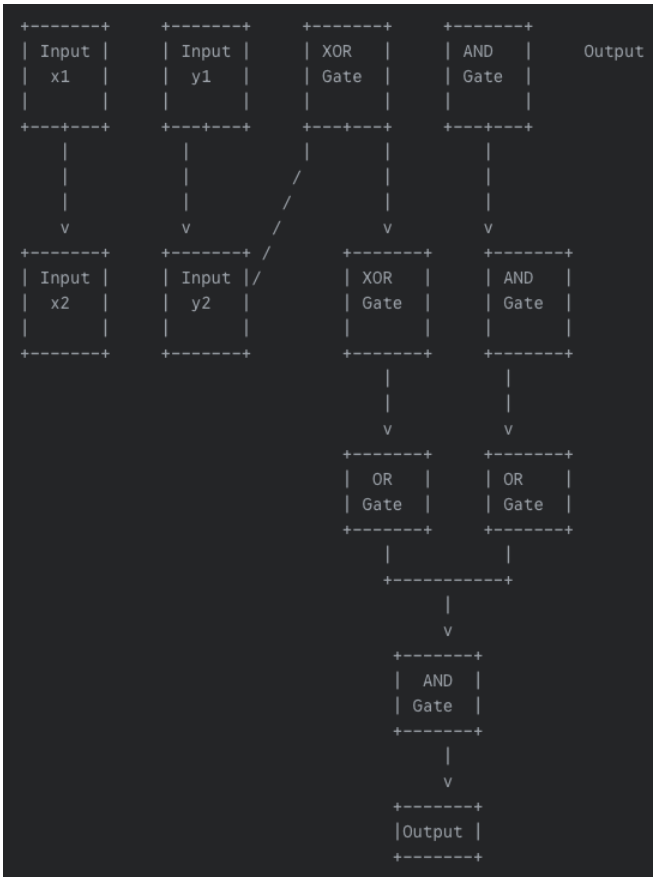
11] M. O. Rabin, "How to Exchange Secrets by Oblivious Transfer," Tech. Rep. TR-81, Aiken Computation Laboratory, Harvard University, 1981.

[12] M. Keller, "MP-SPDZ: A Versatile Framework for Multi-Party Computation," in Proceedings of the 2020 ACM SIGSAC [2] Conference on Computer and Communications Security, pp. 1057-1070, 2020.

[13] S. Zahur, M. Rosulek, and D. Evans, "Two Halves Make a Whole: Reducing Data Transfer in Garbled Circuits using [3] Half Gates," in Advances in Cryptology – EUROCRYPT 2015, D. G. , Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 220–250

Appendix

(A) Example of a Garbled Circuit for String Comparison



A simple example of a garbled circuit for comparing two 2-bit strings

In this circuit:

- x1, x2 are the bits of the first string.
- y1, y2 are the bits of the second string.
- XOR gates are used to compare individual bits.
- AND gates combine the results of the XOR gates.
- The final AND gate outputs 1 if the strings are equal, and 0 otherwise.

This circuit would be garbled by encrypting the truth tables of each gate in a way that allows evaluation but hides the intermediate values.

(B) Example of Secret Sharing

Let's say we want to share a secret value S = 10 among three parties (P1, P2, P3) using Shamir's Secret Sharing with a threshold of 2 (meaning any two parties can reconstruct the secret).

1. **Polynomial Construction:** We randomly choose a polynomial of degree 1 (threshold - 1):
    - f(x) = 10 + 5x (where 10 is the secret and 5 is a random coefficient)
2. **Share Generation:** We generate shares by evaluating the polynomial at different points:
    - P1's share: f(1) = 10 + 5(1) = 15
    - P2's share: f(2) = 10 + 5(2) = 20
    - P3's share: f(3) = 10 + 5(3) = 25

3. **Secret Reconstruction:** Any two parties can reconstruct the secret using Lagrange interpolation. For example, if P1 and P2 combine their shares:
    - S = 15 * (x - 2) / (1 - 2) + 20 * (x - 1) / (2 - 1)
    - At x = 0 (which is where the secret lies in the polynomial), S = 15 * (-2) / (-1) + 20 * (-1) / (1) = 30 - 20 = 10

This example show how Shamir's Secret Sharing allows distributing a secret among multiple parties while ensuring that only authorized subsets of parties can reconstruct it.