iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** XI **Month of publication:** November 2021

DOI: https://doi.org/10.22214/ijraset.2021.38932

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: 7.429
Volume 9 Issue XI Nov 2021- Available at www.ijraset.com

Efficient and Secure Multi-party Computation for Heterogeneous Environment

Surabhi Kumari[1], Avadhesh Kumar Dixit[2], Piyush Rai[3]
[1]*M. Tech Student, Dept. of Computer Science and Engineering, IET. Dr. RMLAU, Ayodhya, U.P, India*
[2]*Assistant Professor, Dept. of Computer Science and Engineering, IET. Dr. RMLAU, Ayodhya U.P, India*
[3]*Assistant Professor, Dept. of Computer Science and Engineering, IET. Dr. RMLAU, Ayodhya U.P, India*

Abstract: ***MPC (multi-party computation) is a comprehensive cryptographic concept that can be used to do computations while maintaining anonymity. MPC allows a group of people to work together on a function without revealing the plaintext's true input or output. Privacy-preserving voting, arithmetic calculation, and large-scale data processing are just a few of the applications of MPC. Each MPC party can run on a single computing node from a system perspective. Multiple parties' computing nodes could be homogenous or heterogeneous; nevertheless, MPC protocols' distributed workloads are always homogeneous (symmetric). We investigate the system performance of a representative MPC framework and a collection of MPC applications in this paper. On homogeneous and heterogeneous compute nodes, we describe the complete online calculation workflow of a state-of-the-art MPC protocol and examine the fundamental cause of its stall time and performance limitation.***
Keywords: Cloud Computing, IoT, MPC, Amazon Service, Virtualization.

I. INTRODUCTION

The rising popularity of distributed computing in recent years has spawned a slew of new projects that make use of a variety of distributed computing platforms. Cloud computing enhances dispersed computing by adding virtualization and administration capabilities, and cloud services have grown in popularity and are widely available from a variety of vendors (e.g. Amazon, Google). Multiple organisations or private persons may provide community clouds.

Finally, new research suggestions include cloud systems that combine in-home, mobile, and data centre computing resources, thereby increasing the overall pool of computing resources by adding the continuously growing computational power from various personal and mobile computing devices. The number of linked devices is rapidly increasing, resulting in the Internet of Things (IoT), a vast network of networks connecting smart devices such as sensors and actuators. Devices like these are used in areas including public health, smart grids, smart transportation, waste management, smart homes, smart cities, agriculture, and energy management, among others.

The requirements and limitations of connected "things" present a number of challenges, including connectivity challenges for billions of devices to communicate with one another, security challenges with the need to protect IoT networks from being attacked (according to a Gartner report, 20% of organisations have experienced at least one IoT attack in the last three years) while also preventing them from being exploited to become an attack tool, and so on (e.g., Mirai botnet). These challenges are “augmented” with the resource-limited nature of IoT devices which renders traditional communication protocols and security schemes inefficient and even infeasible for IoT. The prevalence of IoT devices and their usage in important applications, which amplify the impact of any security breach to the point of becoming life-threatening, are making IoT-related security challenges more alarming. A such scenario might be imagined based on a flaw discovered in a pacemaker device in2017, which led to the recall of 500,000 pacemakers by the US Food and Drug Administration (FDA) due to concerns that security flaws could allow a hacker to manipulate the heart-beat regulating device.

Distributed computing systems can be used for a variety of applications and services, such as media content distribution and editing, content-rich social networks, gaming, data storage, home security, and more. The data in most of these applications could be commercially sensitive or personal, such as family films or photographs acquired by a home security system. This places a high demand on the distributed computing system to handle data in a secure and confidential manner. The growing high level of diversity of distributed systems, which includes diverse computing devices, owners of such devices, and surroundings in which they are operated, strengthens the justification for security and privacy.

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue XI Nov 2021- Available at www.ijraset.com

II. MPC OVERVIEW AND RELATED WORKS

MPC allows several entities to jointly compute a mathematical function that takes inputs from many contributing entities and provides a formal guarantee of the input data's privacy and the computed result's correctness. Yao first presented the two-participant form of MPC as the millionaire's issue. Following that, research efforts will be focused on increasing security and deploying MPC (e.g. SEPIA). Other cryptographic techniques are used in MPC's protocol. Secret sharing is an important strategy that secures the input data's secrecy. Shamir's system, which uses polynomial interpolation, is the most widely applicable secret sharing scheme, which we focus on in this study. Other methods of secret sharing may also be appropriate. Shamir briefly discusses the concept of allocating uneven numbers of shares to participants according on their attributes (e.g., amount of power), which is referred to as a hierarchical scheme. This concept, however, has yet to be implemented in MPC. In this study, we present a strategy for implementing this concept in MPC, and we compare our approach to a baseline mechanism in which each participant receives a single share.

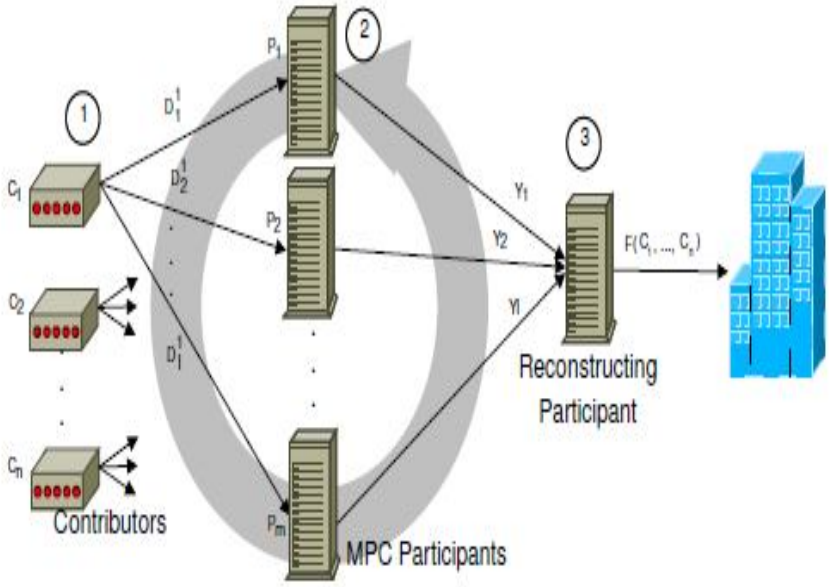


Fig. 1 Illustration of SMPC

An example MPC system using secret sharing is shown in Fig. 1. The MPC protocol consists of the three following stages:

A. Input Sharing

For secure computing, a number of participants offer input data. The input data is exchanged with the participants using the previously stated secret sharing technique. Each secure computing entity may be assigned one or more shares, as shown in Fig. 1.

B. Arithmetic Operation

Using the data they've been given, the participants create a mathematical function. Addition of two values only requires addition of two shares since the two are equivalent. Multiplication of two values is a more complicated process, and after local multiplication of input data shares, re-sharing and reconstruction stages are necessary.

C. Result Reconstruction

This is accomplished by computing a linear combination of the participants' outputs, which is produced using a polynomial interpolation procedure such as Lagrange interpolation or Neville's algorithm.

©IJRASET: All Rights are Reserved

III.MPC IN HETEROGENEOUS ENVIRONMENT

Consider a sensitive data application that is processed on a heterogeneous distributed computing platform with a variety of devices and players operating the devices. Different security standards inside participating corporations or personal (including in-home) computer devices, trust levels in adhering to the established collaboration protocol, and reliability differences of the included devices are all examples of heterogeneity.

TABLE I
Description of variable

k	number of shares required or the threshold to recover the secret
n	total number of contributors (i.e. input data provider)
m	total number of MPC participants
l	total number of shares in the system
r	resolution, describes the relationship between m and l
l_i	number of shares received by participant i from each contributor
c	number of corrupt participants
P_i	participant i
D^j	input from contributor j
D^j_i	share from contributor j's input with index i

As is commonly done in MPC, is likely to result in substandard MPC performance. This motivates our interest in exploring uneven sharing of data and processing in a heterogeneous MPC system. The most obvious consequence of unequal share distribution is that a participant may receive more than a single share. Therefore, together with the potential performance gain, we also need to explore the corresponding processing complexity and communication overhead.

Table 1 lists the variables used in the paper. An MPC system consists of n contributors (that provide the input data Di) and m participants (that perform MPC protocol). An entity can be a contributor, participants or both. Each of the n contributors generates l shares and distributed the shares over m participants, where each participant receives li shares from each contributor. To simplify the analysis, we define resolution, r, the average units of shares per participant which describes the relationship between m and l according to $l = r \times m$. In equal share distribution, $l = m$ and $l_i = 1\ \forall i$ while in unequal share distribution, $l > m$ and $l_i \geq 1\ \forall i \in [1..k-1]$. We also introduce the notion of share index, which refers to the point where the polynomial is evaluated to obtain the share. It is important that the contributors have agreed on l_i that is allocated to each P_i before the protocol begins, to ensure the mathematical function to be properly executed.

The MPC procedure must be updated to allow for the allocation of multiple shares to selected participants and the processing of multiple shares in the case of unequal share distribution. While this work only considers addition and multiplication of two numbers, our method can be applied to other mathematical operations.

For input sharing, unequal share distribution requires contributors to generate a larger number of shares, since $l > m$. Similarly, addition of two values requires each participant to repeat the addition operation l_i times.

Algorithm 1. Multiplication of two values for Pi

Data: Two vectors of shares from two contributors: $[D^\alpha]$ and $[D^\beta]$
Result: Multiplication result vector: $[D^{\alpha\times\beta}]$
for $i \in l_i$ do
  $d_i^{\alpha\times\beta} = D_i^\alpha \times D_i^\beta$
  Generates l shares for $d_i^{\alpha\times\beta} = \{d_{i,1}^{\alpha\times\beta}, ..., d_{i,l}^{\alpha\times\beta}\}$
  for $j = 1$ *to* l *(except for* $j = l_{first}, ..., l_{last}$*)* do
    Sends $d_{i,j}^{\alpha\times\beta}$ to the corresponding participant
  end
end
Computes recombination vector $[r] = [r_1, ..., r_l]$ using polynomial interpolation
for $i \in l_i$ do
  Recovers the final multiplication results: $D_i^{\alpha\times\beta} = \sum_{j=1}^{l} r_j \times d_{j,i}^{\alpha\times\beta}$
end

©IJRASET: All Rights are Reserved

We present details of the multiplication of two input values for unequal share distribution MPC in Algorithm 1. First, each participant needs to locally multiply two shares with the same share index for each of the l_i shares. Each participant then re-share each local multiplication result to all other participants. In the final step, each participant needs to reconstruct the result. To do this, the participant computes the recombination vector, [r] using interpolation to solve the polynomial of size l. The final result for each l_i is then recovered by summing up $d_{1,i}^{\alpha\times\beta}, \ldots, d_{l,i}^{\alpha\times\beta}$ with [r] as weights.

For result reconstruction in unequal distribution MPC, each participant must send l_i outputs to the reconstructing entity. The recombination vector computation and result recovery are equivalent to the operations in MPC with equal share distribution.

IV. PERFORMANCE EVALUATION

The benefit of unequal distribution of shares Simulations based on the original C++ implementation are used to evaluate MPC. Because determining the ideal share distribution for uneven MPC is fundamentally a combinatorial optimisation problem with no analytical solution, we did not do an analytical study.

1) *Simulation, Results and Discussion:* We consider a scenario where the participants can be divided into two classes: resilient participants which have low P_{ci} (close to fully honest) and vulnerable participants which have high P_{ci} (close to corrupt). We note that this represents the upper bound on the potential performance gains compared to the unequal share distribution, and that considering participants with closer corrupt probabilities may result in lower performance improvement. We present only the results for MPC integrity failure that is for $k = \frac{l}{3}$ for privacy failure only relates to a threshold change. In the simulations, we vary m, r and the share distribution algorithm. m is increased from 6 to 21, higher m is not simulated due to high complexity. We use r = 2, 5, 10 for unequal share distribution MPC, where r = 2 represents the smallest possible resolution and r = 5, 10 represent higher resolutions. Note that r can be any positive integer. The performance of equal share distribution is considered as a baseline, which we compare to the performance of the exhaustive search, heuristic method and GA share distribution algorithms. For resilient participants P_{ci} = 0.1 and for the remaining participants P_{ci} = 0.9.

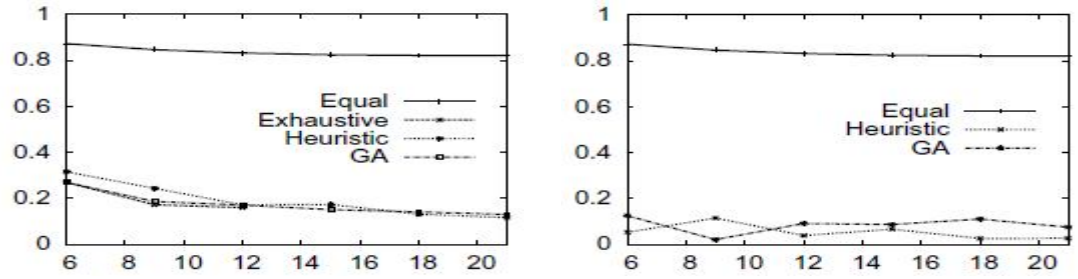


Fig. 2 Failure probability of MPC with different share distribution algorithms for resolution 2 and 5 in the scenario where a third of the participants are vulnerable.

Fig. 2(a) and 2(b) present the failure probability results for r = 2 and r = 5. We can see that *pfail* for unequal share distribution MPC is significantly lower regardless of the share distribution algorithm used. Due to high complexity, exhaustive search has been omitted for the subsequent evaluation, but it is apparent from Fig. 2(a) that the performance of GA (and the heuristic method for m = 12) closely follows exhaustive search. Fig. 2(b) shows the result for r = 5. In this case, heuristic method performs slightly better than GA. The graph for r = 10 is not included as the result is similar to r = 5 in this particular scenario.

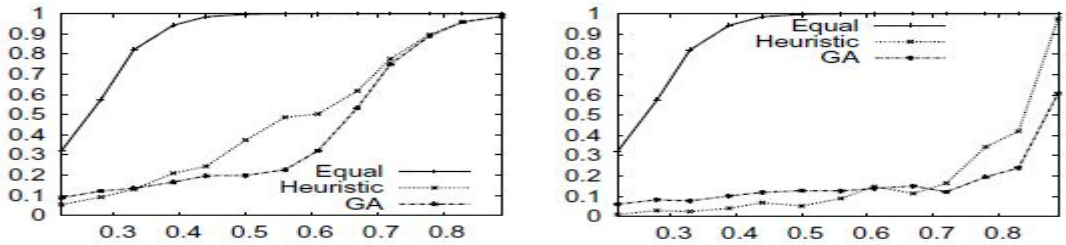


Fig. 3 . Failure probability of MPC when the total number of participants is 18 andthe number of vulnerable participants is varied between 4 and 16

©IJRASET: All Rights are Reserved

The performance improvement of unequal share distribution is demonstrated in Fig. 3(a) and Fig. 3(b). We can observe that the failure probability of equal distribution MPC rapidly increases as the percentage of corrupt participants increases. Complete failure (i.e. P_{fail} = 1) for equal share distribution occurs when at least 45% of the participants are vulnerable. On the other hand, unequal share distribution MPC suffers complete failure when there are close to 90% vulnerable participants in the system. The resolution only affects the rate of change of P_{fail}, where the rate of change is higher when r = 2 compared to when r = 5. As can be logically expected, this demonstrates the improved resilience of unequal share distribution compared to equal distribution MPC to the number of low trustworthy participants.

While we evaluate a specific scenario, this result can be applied to a more generic case where each participant has different P_{ci}.

V. NUMERICAL EVALUATION

We numerically evaluate the group computation and communication overhead, defined as the ratio of the complexity of the unequal distribution MPC to the equal distribution MPC, i.e. the increase in operations or messages required. m is varied between 6 and 120, and for unequal MPC r= 2, 5, 10. Fig. 4(a) to 4(d) shows the overhead of unequal distribution MPC as m increases. In all the graphs, the ratio converges to a value, which can be obtained by dividing the complexity of unequal distribution MPC to equal distribution MPC. For input sharing, the group overhead ratio is represented by Fig. 4(a), where the ratio approaches r^2 as m increases. Fig. 4(b) shows the overhead ratio for addition of two values. We can observe that the ratio is equal to r, which is expected from the results. The group overhead for multiplication of two values is presented in Fig. 4(c). The trend shows that the ratio approaches $\frac{r3+r2}{2}$ for large m. Finally, Fig. 4(d) presents the overhead ratio for result reconstruction. Similar to input sharing, the ratio approaches r^2 when m increases.

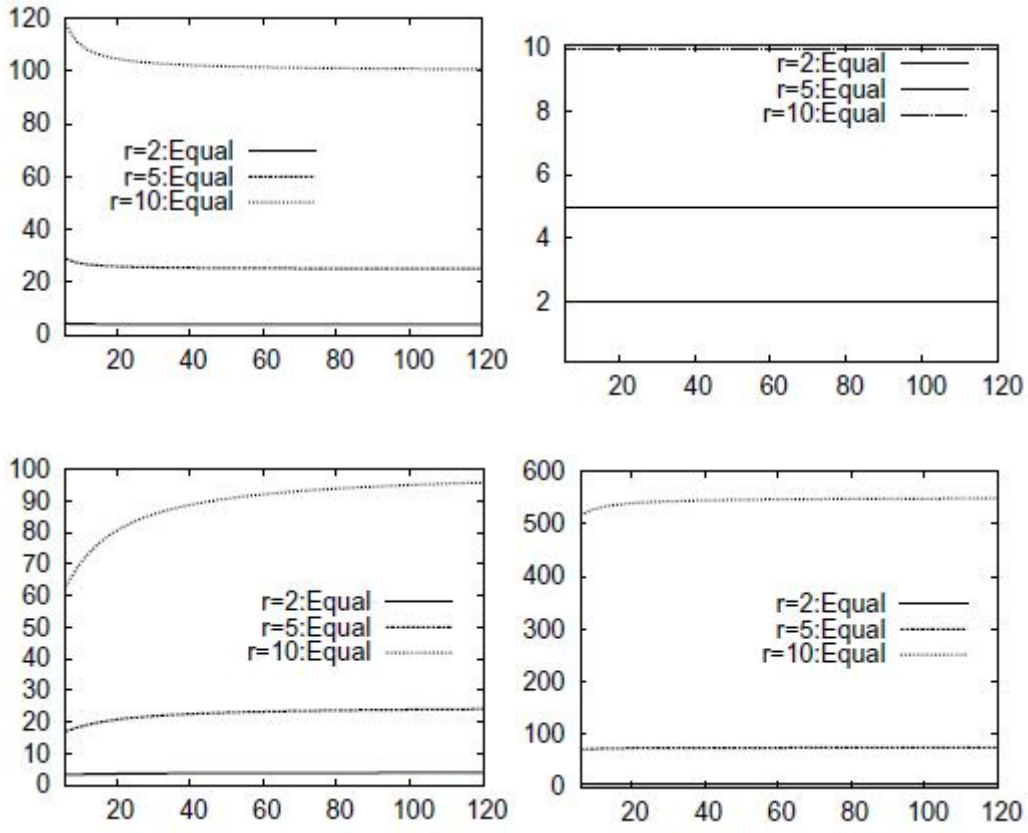


Fig. 4 . Group computational overhead in terms of the ratio of the computational complexity of the unequal distribution MPC to the equal distribution MPC

The communication complexity in Table 3 shows that the ratio for input sharing and result reconstruction is proportional to r while for multiplication of two values the overhead ratio is r^2.

©IJRASET: All Rights are Reserved

VI. DISCUSSION

We calculate the time it takes for MPC operations on mobile devices to see if unequal share distribution is feasible. The MPC operation of multiplication of two values is chosen because it has the highest computational complexity of all the MPC procedures. The estimated computation time presented in Table 2 is computed by using the expressions in Table 1 and the computation power of the mobile device obtained using Linpack benchmark (e.g., Android Nexus S device has the computation power of 17 MFLOPS). While the MPC operations may not always be floating point, this provides the upper bound on the expected values as the floating point operations require more time to compute than integer operations.

TABLE III
Estimated computation time for the multiplication of two values (MS)

m\r	1	2	5	10
6	0.0046	0.027	0.33	2.36
12	0.018	0.10	1.29	9.39
30	0.11	0.64	8.0	58.41
60	0.43	10.19	127.24	932.47

The system is still considered to be practical if the computation time is below or close to the Internet latency. Assuming that MPC participants are located worldwide, the average latency is estimated to be 100 ms. As can be seen from Table, the computation time for small *m* is still well below the average latency even for high resolution. The computation time becomes very high when both *m* and *r* are high. The increase in communication load is considered less critical than the complexity, as the size of MPC protocol messages is small. Although the full impact of unequal distribution MPC will depend on the protocol implementation and the computing platforms, we have clearly demonstrated the improved performance of unequal distribution compared to that of equal distribution on heterogeneous platforms. While the computation and communication overheads increase with *r*, it is not always the case for the robustness. Hence, it is important to carefully select the value of *r*.

VII. CONCLUSIONS

We investigate unequal share distribution in MPC on heterogeneous platforms and show that it improves the system's robustness greatly when compared to the typical situation in which all players have the same number of shares. However, such progress is accompanied by an increase in complexity and overhead. As a result, when determining parameter values, we take into account the trade-off between the two. We intend to examine the methods for reducing the complexity and costs of unequal share allocation in MPC in the future.

VIII. ACKNOWLEDGMENT

I Surabhi Kumari, student of Master of Technology, Greatful to IET Dr. Rammanohar Lohia, Avadh University, Ayodhya. I have completed my Research Paper under the supervision of my Guide Er Avadhesh Kumar Dixit and Co-Guide Er. Piyush Rai.

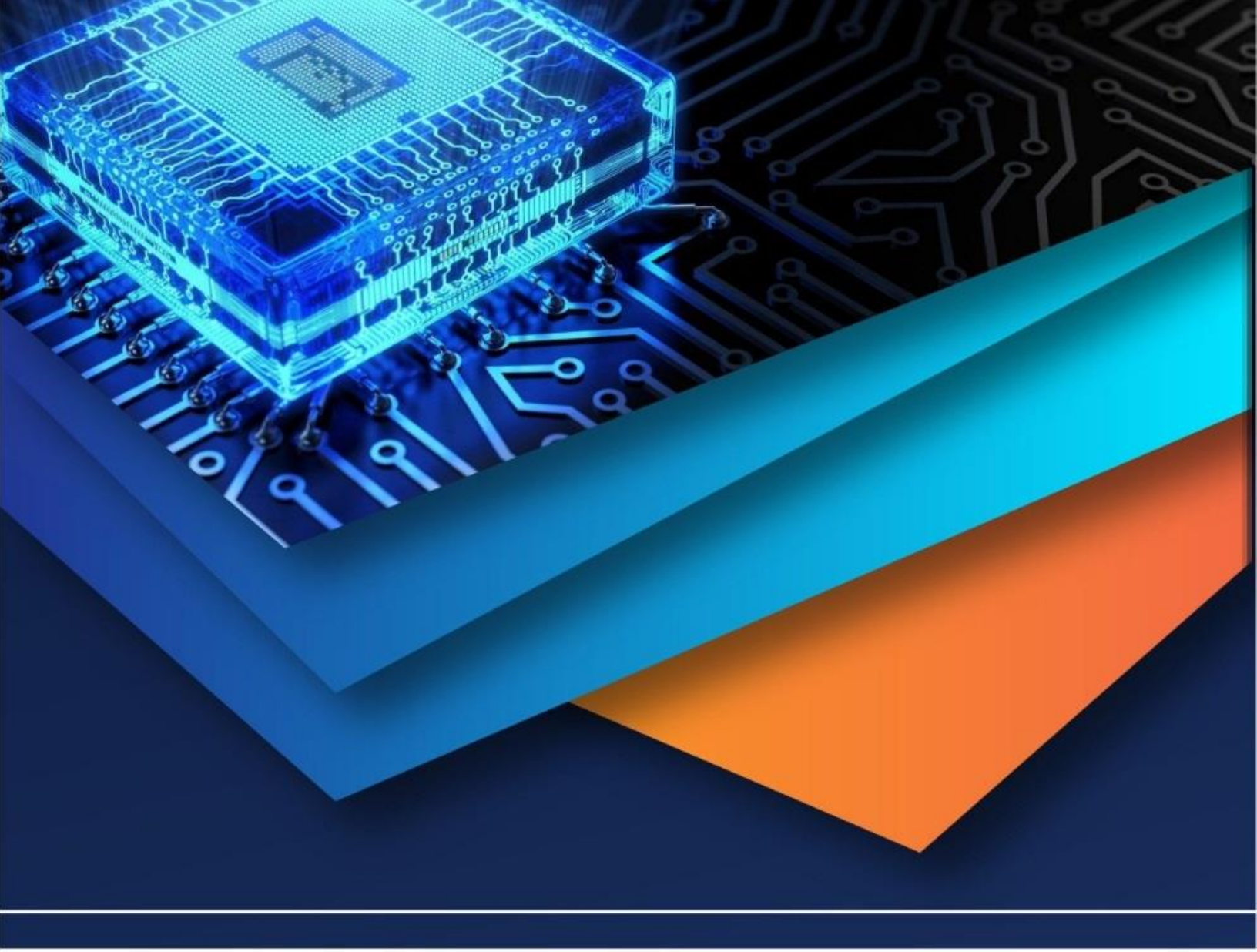
REFERENCES

[1] Zhao, Chuan, et al. "Secure multi-party computation: theory, practice and applications." Information Sciences 476 (2019): 357-372.
[2] Mohanta, Bhabendu Kumar, Debasish Jena, and Srichandan Sobhanayak. "Multi-party computation review for secure data processing in IoT-fog computing environment." International Journal of Security and Networks 15.3 (2020): 164-174.
[3] Sarhan, Akram Y., and Steve Carr. "A highly-secure self-protection data scheme in clouds using active data bundles and agent-based secure multi-party computation." 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud). IEEE, 2017.
[4] Du, Wenliang, and Mikhail J. Atallah. "Secure multi-party computation problems and their applications: a review and open problems." Proceedings of the 2001 workshop on New security paradigms. 2001.
[5] Knott, Brian, et al. "CrypTen: Secure multi-party computation meets machine learning." arXiv preprint arXiv:2109.00984 (2021).
[6] Kamara, Seny, Payman Mohassel, and Mariana Raykova. "Outsourcing Multi-Party Computation." IACR Cryptol. Eprint Arch. 2011 (2011): 272.
[7] Tso, Raylin, et al. "Privacy-preserving data communication through secure multi-party computation in healthcare sensor cloud." Journal of Signal Processing Systems 89.1 (2017): 51-59.
[8] Wu, Yulin, et al. "Generic server-aided secure multi-party computation in cloud computing." Computer Standards & Interfaces 79 (2022): 103552.
[9] Das, Debasis. "Secure cloud computing algorithm using homomorphic encryption and multi-party computation." 2018 International Conference on Information Networking (ICOIN). IEEE, 2018.
[10] Klinec, Mgr Dušan. "Secure multi-party computation with resource-constrained devices."
[11] Wolfe, Pierre-François W. Enabling secure multi-party computation with FPGAs in the datacenter. Diss. Boston University, 2021.
[12] Ankele, Robin, and Andrew Simpson. "On the performance of a trustworthy remote entity in comparison to secure multi-party computation." 2017 IEEE Trustcom/BigDataSE/ICESS. IEEE, 2017.

©IJRASET: All Rights are Reserved

[13] Harn, Lein, Zhe Xia, and Chingfang Hsu. "Non-interactive secure multi-party arithmetic computations with confidentiality for P2P networks." Peer-to-Peer Networking and Applications 14.2 (2021): 722-728.
[14] Lakshmi, C. "AN EFFICIENT SECURE COMPUTATION FOR PRIVACY PRESERVING DATA MINING IN MULTI PARTY COMPUTATION (MPC)–A." Technology 11.10 (2020): 855-870.
[15] Wang, Zhaohong, Sen-Ching S. Cheung, and Ying Luo. "Information-theoretic secure multi-party computation with collusion deterrence." IEEE Transactions on Information Forensics and Security 12.4 (2016): 980-995.
[16] Sarhan, Akram Y. Protecting Sensitive Data in Clouds Using Active Data Bundles and Agent-Based Secure Multi-Party Computation. Diss. Western Michigan University, 2017.
[17] Shukla, Samiksha, and G. Sadashivappa. "Secure multi-party computation protocol using asymmetric encryption." 2014 International Conference on Computing for Sustainable Global Development (INDIACom). IEEE, 2014.
[18] Yigzaw, Kassaye Yitbarek, and Johan Gustav Bellika. "Evaluation of secure multi-party computation for reuse of distributed electronic health data." IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI). IEEE, 2014.
[19] Oleshchuk, Vladimir A., and Vladimir Zadorozhny. "Secure multi-party computations and privacy preservation: Results and open problems." Telektronikk 103.2 (2007): 20.
[20] Stammler, Sebastian, et al. "Mainzelliste SecureEpiLinker (MainSEL): Privacy-Preserving Record Linkage using Secure Multi-Party Computation." Bioinformatics (2020).
[21] Mou, Wenhao, et al. "A Verifiable Federated Learning Scheme Based on Secure Multi-party Computation." International Conference on Wireless Algorithms, Systems, and Applications. Springer, Cham, 2021.
[22] Huang, Xueqing, and Nirwan Ansari. "Secure multi-party data communications in cloud augmented IoT environment." 2017 IEEE International Conference on Communications (ICC). IEEE, 2017.
[23] Niu, Shufen, Zhenbin Li, and Caifen Wang. "Privacy-preserving multi-party aggregate signcryption for heterogeneous systems." International Conference on Cloud Computing and Security. Springer, Cham, 2017.
[24] Lopez-Fenner, Julio, et al. "Privacy Preserving Multi Party Computation for Data-Analytics in the IoT-Fog-Cloud Ecosystem." CICCSI 2020: IV International Congress of Computer Sciences and Information Systems. 2020.
[25] Yigzaw, Kassaye Yitbarek, et al. "Towards privacy-preserving computing on distributed electronic health record data." Proceedings of the 2013 Middleware Doctoral Symposium. 2013.

©IJRASET: All Rights are Reserved



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)