Security Preserving Distributive n-Party Computation*

Mou Dutta[1], Subhasish Dhal[2], Ashi Saxena[3] and Mahasweta Sarkar [4]

***Abstract*—In n-party Computation, data from different sources are integrated to achieve a common goal, which can be beneficial for different applications. However, this goal may be at risk due to the fact that this corresponding data can be the target of attacks from outside attackers as well as inside ones. On the other hand, in Secure n-party Computation, the data can be processed by preserving security and privacy in such a way that no party can know the data of the others. Several schemes have been proposed to address the issues regarding n-party computation. However, some of them are not efficient in terms of computation and communication, or some of them have used a Trusted Initializer (TI), a third party, which is considered as trusted, for communicating between different parties. Therefore, to address these issues, we propose an n-party secure computation protocol without using any Trusted Initializer (TI). We formally simulate our protocol using Automated Verification of Internet Security Protocols and Applications (AVISPA) as well as we elaborate on the formal and informal security analysis of our proposed protocol. Performance analysis of our proposed protocol is also carried out and it is observed our proposed protocol is secure and protects the privacy of the system.**

***Index Terms*—n-party computation, secure n-party computation, trusted initializer**

I. Introduction

Secure n-party Computation also known as Secure Multi-party Computation, Secure Computation, Multi-Party Computation (MPC), or privacy-preserving computation [1], [2] is one of the most active research areas in both theoretical and applied cryptography [3]. The aim is to correctly compute some information using the information of two or more parties while keeping all the information of a party secret from the other parties. There are many applications of n-party computation (MPC) such as secure machine learning, Brain Computer Interface (BCI), and others.

The parties involved in the computation are never acquainted with the sensitive input of the other party, which is crucial. The idea of MPC was first introduced in the early 1980s by Andrew Yao [4] with the millionaires' problem and was intensively studied for decades. Recently, MPC has become efficient enough to be used in practice and has made the transition from being studied theoretically to a technology being used in industry [5]. Secure voting [6], [7] secure machine learning [8], brain-computer interfaces [9], privacy-preserving network security monitoring [10], etc. are only a few of the many applications of Secure n-party Computation. Despite these significant facts, secure n-party computation is still not considered feasible enough for use in the bulk of applications that require (near) real-time performance [11], [12].

Numerous works [4], [13]–[18] in the area of MPC have been performed over the decades. These works, on the other hand, contain different forms of limits in terms of computation to different powerful attacks, including a third party, which they have assumed as a trusted entity, sometimes referred to as a "trusted initializer" (that can be compromised at any time), or has been limited between two parties, and so on. Suppose, Alice and Bob have the shares of X and Y and their aim is to compute XY in a way that their shares can be hidden from each other. Here, $XY{=}(X_{Alice} + X_{Bob})(Y_{Alice} + Y_{Bob}){=}X_{Alice}Y_{Alice}{+}X_{Alice}Y_{Bob} + X_{Bob}Y_{Alice} + X_{Bob}Y_{Bob}$. The expressions $X_{Alice}Y_{Alice}$ and $X_{Bob}Y_{Bob}$ can be locally computed by Alice and Bob themselves. However, the computation of $X_{Alice}Y_{Bob}$ and $X_{Bob}Y_{Alice}$ are more complicated due to the fact that no party wants to share their parts with the other party. To address this issue, authors of [27] have presented a protocol for matrix multiplication between two parties. However, in their protocol, they have used a third party or in other words, an initializer for the communication between the two parties, which they have assumed to be trusted. Relying on a trusted initializer has the disadvantage that while it is considered to be trustworthy, it may also be curious, which might result in undesirable or unexpected consequences. As it strictly follows predefined protocols, it may have a direct interest in inferring sensitive information from interactions between various parties. In another work, to address the aforementioned issue, authors of [19] have proposed a protocol for secure multiplication between two parties without a trusted initializer. Their protocol has been restricted to only two parties. In addition to that, they have not addressed the most important attacks like man-in-the-middle, replay, impersonation attacks, etc., which can make the system computationally vulnerable. For instance, in the case of applications involving brain-computer interfaces, data from a large number of parties are needed for training, however, no party wants their sensitive data to be made public to the other parties. Because a malicious party may have harmful effects on the data owner if they gained access to those sensitive data. In order to address this issue, we develop a generalized communication protocol for n parties without a trusted initializer in this paper. In our proposed protocol, we use different methods to make our protocol secure against some well-known attacks such as replay attack, man-in-the-middle attack, impersonation attack and our protocol also preserves the integrity of the data. This

*This research is funded by the Science and Engineering Research Board (SERB) [File number EEQ/2020/000039], Department of Science and Technology, Government of India.

[1]Mou Dutta is with Department of Computer Science & Engineering, Indian Institute of Information Technology Guwahati, Assam, India mou.dutta@iiitg.ac.in

[2]Subhasish Dhal is with the Department of Computer Science & Engineering, Indian Institute of Information Technology Guwahati, Assam, India subhasis@iiitg.ac.in

[3]Ashi Saxena is with the Department of Computer Science & Engineering, Dr. Akhilesh Das Gupta Institute of Technology and Management, India ashi.saxena.888@gmail.com

[4]Mahasweta Sarkar is with the Department of Electrical & Computer Engineering, San Diego State University, United States of America msarkar2@sdsu.edu

paper has the following contributions:

I. Through Secure n-party Computation, data from two or more parties need to be incorporated to generate an output without revealing their data to each other. Here, we consider a scenario where n-parties (each with their own data) want to perform a computation by integrating all data by directly interacting among themselves without involving any third party (such as TI), and without exposing their data to each other. Our proposed protocol is capable of maintaining data integrity and confidentiality while also being resistant to impersonation attacks, replay attacks, man-in-the-middle attacks, etc., which as per our knowledge, the existing works on this domain have been failing to address.

II. We propose homomorphic encryption based n-party secure computation protocol, as it allows the system to do computation over encrypted data without decrypting it. The main purpose of using homomorphic encryption is to make it possible to compute on encrypted data. As a result, data can stay private while being analyzed, allowing beneficial tasks to be completed with data stored in an untrustworthy environment.

III. Security of our proposed protocol has been formally and informally analyzed and our protocol has been simulated using the well-known formal verification tool Automated Verification of Internet Security Protocols and Application (AVISPA) [34]. We also carry out the performance analysis of our proposed protocol in terms of computation, security, etc.

The outline of the paper is as follows: Section II summarizes the related works related to secure n-party computation. Section III presents the preliminaries related to the proposed work, followed by the proposed work in Section IV. The security and privacy analysis have been presented in Section V and the performance analysis has been presented in Section VI, followed by the conclusion in Section VII.

II. Related Works

Secure n-party Computation was first proposed by A. Yao [4] in 1982 and then it was extended by Goldreich *et al.* [13]. The growing fields of MPC are based on Oblivious Transfer [14] and retrieval of private information from cloud [15]. Oblivious Transfer is a cryptographic protocol, which allows a sender to communicate a portion of its inputs to a chooser in such a way that the chooser does not acquire more information than it is authorized to and the sender does not know which part of the inputs, the chooser obtained. In [16], Shamir has proposed a method where a set of data D can be divided in n parts in such a way that D can be reconstructed using at least k pieces, however, no information about D can be revealed by $k-1$ pieces or less. Authors of [17] had presented a new construction of a highly efficient 1-out-of-N Oblivious Transfer where only O(Log N) execution time is needed for 1-out-of-2 Oblivious Transfer Protocol. They also presented k-out-of-N Oblivious Transfer in the same paper, which is more efficient than k folds of 1-out-of-N Oblivious Transfer, where the receiver obtains only one element out of N elements without the sender knowing which element was queried as well as the receiver didn't get to know about the other elements that were not queried. In an Oblivious Transfer, the chooser obtains the one object he has chosen while the sender maintains n items. The object that was transferred is unknown to the sender. This could be a problem if each party's data is completely required for computation and obtaining the best result. Nielsen *et al.* [18] have presented a practical approach for two-party computation that was secure against an active adversary. In this paper, an Oblivious Transfer-based approach using a random oracle model had been introduced by them along with a number of new techniques for related inputs and outputs in an Oblivious Transfer for larger constructions. The authors in [19] have also presented a secure multiplication protocol where one party doesn't want to share their data with another party. However, their proposal is limited to two parties only, and they didn't provide any formal or informal security analysis of their proposed protocol. Moreover, they have not addressed some of the well-known attacks like man-in-the-middle, replay, impersonation, etc.

The two-party secret sharing protocol based on additive secret sharing has been introduced by Pullonen *et al.* [20], where they have implemented multiplication protocol using secret sharing. T. Geng *et al.* [21] have also employed a secret sharing mechanism in blockchain technology to address the issue of low-efficiency and high-cost consensus protocol. Author in [23] has also mentioned the use of secret sharing for their cheater detection protocol. In [22], authors have used additive secret sharing-based secure n-party computation for implementing their optimized CNN. However, these secret sharing mechanisms may fail due to reasons that there is a third party involved, who gathers the shares from both parties, recovers the secret value, and then provides new shares of the value to both parties in an evenly divided manner for their resharing process. They have made the supposition that all conversations have taken place via a secure channel in their multiplication protocol.

The scheme, presented by Atallah *et al.* [24], has been beneficial for small organizations. By using this scheme, they can cooperatively work without revealing their data to others, as by sharing with each party their desired queried answers without revealing the data of the other participant. However, they have not mentioned any mechanism to address the MITM, replay, or impersonation attack which can compromise the whole system. Authors of [25] have proposed a method for n-party secure computation by secret sharing that has been used for secure integer and fixed-point operations. The authors in [26] have presented the same for rational numbers. The author in [27] presented a one-time table for two-party secure computation, where a third party or trusted initializer was also involved in the system. The authors in [28] have utilized a Secure Multiplication Protocol, SMul, in which, they have used a third server, which is considered as a trusted party. In their protocol, from the data processed through communication, trusted third server may know the inputs of the participants of the protocol. The proposed work in [29] can suffer from man-in-the-middle attack, replay attack, etc. In [30], authors have proposed an anonymous authentication protocol for privacy-preserving distributed learning, where they have also used a third party,

TABLE I
DESCRIPTION OF NOTATIONS

Symbol	Definition
X_i, Y_i	X and Y's share of i^{th} party
$E()$	Encryption Function
$D()$	Decryption Function
C_{X_i}	Encrypted X_i of i^{th} party
C_{Y_i}	Encrypted Y_i of i^{th} party
$H()$	Hash Function
T_i	Time generated by i^{th} party
T_k	Time generated by k^{th} party
$\|\|$	Concatenation Function
S_i, S_k	Signature generated by i^{th} and k^{th} party

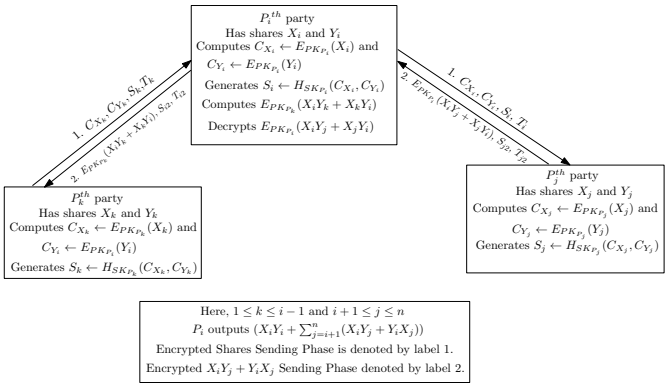

Fig. 1. Secure Multi-party Computation for P_i^{th} Party

which they assume to be trusted.

To address the aforementioned issues, we propose a protocol for n-party computation where no party has to share or reveal their share to another party. Our protocol can be applied in any system involving n number of parties.

III. PRELIMINARIES

This section provides a quick overview of the homomorphic encryption and adversary model.

A. Homomorphic Encryption

Secure n-party computation distributes computation among multiple parties when none of the participants desire to share their own data. Without ever moving or exposing the data, it can allow analysts and data scientists to compliantly and securely compute on remote data. The homomorphic encryption technology, which enables computation on encrypted data without any type of decryption, can be used in this case. The data has always been encrypted once until the entire computation is finished, and the associated private key is the only way to decrypt it. In our proposed protocol, each party does not want their share to be discernible to another party. Hence, in our proposed protocol, we leverage the homomorphic encryption properties in Paillier's encryption scheme [31], [32]. For our work, the following homomorphic encryption (here, PK is the corresponding public key used for encryption) properties have been used.

$$E_{PK}(x).E_{PK}(y) = E_{PK}(x+y) \tag{1}$$

$$(E_{PK}(x))^y = E_{PK}(xy) \tag{2}$$

B. Element-wise Homomorphic Encryption

In this subsection, we discuss the procedure of homomorphic encryption on a matrix where each element of the matrix is encrypted individually [19]. Here, for a matrix B, $b[p, q]$ denotes the element of pth row, qth column and C is the product of the multiplication.

$$C[p,r] = \prod_{p=1}^{d_1}\prod_{r=1}^{d_3}\prod_{q=1}^{d_2}\left({a_{A[p,q]}}^{E_{PK}b_{B[q,r]}}.E_{PK}{a_{B[p,q]}}^{b_{A[q,r]}}\right) \tag{3}$$

IV. PROPOSED WORK

In this section, we present our proposed protocol along with the system model and threat model.

A. System Model

In our proposed model, there exist n parties such as P_1, P_2, P_3, ..., P_n, who are responsible for communication among each other. Each party has its shares of X and Y. We assume that all the participating parties are honest but curious throughout the computation.

B. Adversary Model

To define the capability of the adversary, we use the widely known Dolev-Yao threat model. An adversary $\mathcal{A}$ in a "Dolev-Yao (DY) adversary model" [33] is allowed to intercept all transmitted information between the parties and can alter or destroy the messages. Each partner P_i in our model is supposed to be an untrustworthy and curious entity. As a result, each party may desire to know the shares of the other parties, which could pose a privacy and security risk to our protocol.

C. Overview of the Proposed Protocol

Suppose there are n parties who want to distributively compute XY where (X_i, Y_i) are the shares of party P_i $[1 \le i \le n]$. XY has to be computed distributively which is equal to $(X_1+X_2+X_3+...+X_n)\times(Y_1+Y_2+Y_3+...+Y_n)$. All the public keys, PK_{P_i} are distributed through secure channels. Let X and Y are the matrices of size $d_1 \times d_2$ and $d_2 \times d_3$, respectively. Two parties compute multiplication operation on their shares of matrices using Equation 3. In this protocol, one party A encrypts its matrix element-wise and send that to another party B. Then B merges its shares using Equation 1, Equation 2, and Equation 3.

D. Proposed Protocol in Detail

There are two phases in our protocol namely Initialization Phase and Distributed Multiplication Phase. In the initialization phase, each participating party generates their public-secret key pair and distributes their public keys. In the distributed multiplication phase, all the parties send their encrypted shares to all the parties and distributively complete the task of multiplication.

Algorithm 1 Initialization Phase for Party P_i

1) Party P_i generates its public and secret key pair (PK_{P_i}, SK_{P_i}).
2) P_i distributes its public key PK_{P_i} to other parties through secure channels.

Algorithm 2 Multiplication Phase for ith Party

Input: Shares of X and Y, say (X_1, Y_1), (X_2, Y_2),...,(X_n, Y_n) **Output:** The value of XY

1) P_i encrypts its shares X_i and Y_i using its public key PK_{P_i} and generates two ciphertexts $C_{X_i} \leftarrow E_{PK_{P_i}}(X_i)$ and $C_{Y_i} \leftarrow E_{PK_{P_i}}(Y_i)$.
2) P_i generates its signature $S_i \leftarrow H_{SK_{P_i}}((C_{X_i}, C_{Y_i})||T_i)$, and sends C_{X_i}, C_{Y_i} along with S_i to $P_{i+1}, P_{i+2}, ..., P_n$.
3) P_i obtains C_{X_k}, C_{Y_k}, S_k and T_k from P_k, [where, $1 \leq k \leq i-1$] and verifies
   If $T_k - T_i < \Delta t$ and $S_k = H'_{PK_{P_i}}((C_{X_k}, C_{Y_k})||T_k)$ **Then**

   P_i computes $E_{PK_{P_j}}(X_iY_k + X_kY_i)$ using Equation 1 and Equation 2 and computes signature $S_{i2} \leftarrow H_{SK_{P_i}}((E_{PK_{P_j}}(X_iY_k + X_kY_i))||T_{i2})$ on it and sends signature along with $E_{PK_{P_j}}(X_iY_k + X_kY_i)$, T_{i2} to P_k, [where $1 \leq k \leq i-1$].

   P_i verifies the signature $S_{j2} \leftarrow H'_{PK_{P_j}}((E_{PK_{P_j}}(X_iY_k + X_kY_i))||T_{i2})$, checks T_{i2} and decrypts $X_jY_i + X_iY_j$ upon successful verification [where $i+1 \leq j \leq n$] using its private key upon successful verification and outputs $(X_iY_i + \sum_{j=i+1}^{n}(X_iY_j + X_jY_i))$.

1) Initialization Phase: Algorithm 1 initializes the system for party P_i. In the first step of this phase, party P_i generates its public and secret key pair (PK_{P_i}, SK_{P_i}). Then, in Step 2, P_i distributes its public key PK_{P_i} to other parties in the system through secure channels.

2) Distributed Multiplication Phase for i^{th} Party: This phase describes how Party P_i communicates with other parties. At first step, P_i encrypts its shares X_i and Y_i using its public key PK_{P_i} and generates two ciphertexts $C_{X_i} \leftarrow E_{PK_{P_i}}(X_i)$ and $C_{Y_i} \leftarrow E_{PK_{P_i}}(Y_i)$. Then in Step 2, P_i generates its signatures $S_i \leftarrow H_{SK_{P_i}}((C_{X_i}, C_{Y_i})||T_i)$, and sends C_{X_i}, C_{Y_i} along with S_i to P_j [where, $i+1 \leq j \leq n$]. In Step 3, P_i first obtains C_{X_k}, C_{Y_k}, S_k from P_k [where $1 \leq k \leq i-1$] and then checks whether S_k is equal to $H'_{PK_{P_i}}, ((C_{X_k}, C_{Y_k})||T_k)$ and $T_k - T_i < \Delta t$ [Where, T_k, T_i and Δt are the timestamps of the message received, timestamp of the sent message and a predefined threshold value, respectively]. If both conditions, $T_k - T_i < \Delta t$ and $S_k = H'_{PK_{P_i}}((C_{X_k}, C_{Y_k})||T_k)$ are satisfied, then P_i computes $E_{PK_{P_j}}(X_iY_k + Y_iX_k)$ using Equation 1 and Equation 2, computes signature $S_{i2} \leftarrow H_{SK_{P_i}}((E_{PK_{P_j}}(X_iY_k + X_kY_i))||T_{i2})$ on it and sends signature along with $E_{PK_{P_j}}(X_iY_k + X_kY_i)$, T_{i2} to P_k, [where $1 \leq k \leq i-1$]. Then, P_i verifies the signature $S_{j2} \leftarrow H'_{PK_{P_j}}((E_{PK_{P_j}}(X_iY_k + X_kY_i))||T_{i2})$, checks T_{i2} and decrypts $X_jY_i + Y_jX_i$ [where $i+1 \leq j \leq n$] using its private key SK_{P_i}. For example, party P_i obtains encrypted values of $X_iY_{i+1} + X_{i+1}Y_i$, $X_iY_{i+2} + X_{i+2}Y_i$, $X_iY_{i+3} + X_{i+3}Y_i$,..., $X_iY_n + X_nY_i$ from party P_{i+1}, P_{i+2}, P_{i+3},..., P_n, respectively. Therefore, party P_i outputs $(X_iY_i) + \sum_{j=i+1}^{n}(X_iY_j + Y_iX_j)$. The communication of our proposed protocol is illustrated in Fig. 1.

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/5PartyProtocol.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 128.62s
  visitedNodes: 23192 nodes
  depth: 21 plies
```

Fig. 2. Avispa Simulation Result using OFMC Backend

V. Security and Privacy Analysis

The robustness of our proposed protocol has been analyzed in terms of privacy and security here.

A. Protocol Simulation using AVISPA

Automated Verification of Internet Security Protocols and Applications (AVISPA) [34] is a well-known verification tool that is used to perform the simulations of the security protocols using High Level Protocol Specification Language (HLPSL). We simulate our proposed protocol using AVISPA tool. Five roles are involved in this simulation such as A, B, C, D, and E, which are the parties involved in distributed communication and they own (*X_A, Y_A*), (*X_B, Y_B*), (*X_C, Y_C*), (*X_D, Y_D*), and (*X_E, Y_E*), respectively. The security and privacy goals that we define in our simulation are as follows, i) the preservation of the privacy of the data, ii) the authentication of the transmitted data. Additionally, in role *environment*, we set *sha256* as a hash_func; sec_1, auth_1 as *protocol_id*; *a, b, c, d, e, ka, kb, kc, kd, ke, ki, inv(ki)* as *intruder_knowledge*, where a, b, c, d, e are five parties; *ka*, *kb*, *kc*, *kd*, *ke* are the public keys of a, b, c, d, e, respectively; and *ki*, *inv(ki)* are the public and private keys of intruder *i*. Fig. 2 shows the result of our analysis that our proposed protocol is 'SAFE' in On-the-fly Model-Checker (OFMC).

B. Informal Security Analysis of the Proposed Protocol

This subsection provides the informal security analysis of our proposed protocol.

***Claim* 1:** The proposed protocol preserves confidentiality.

Proof: This proof demonstrates that the secrecy of each party's data is being reserved. The public key E_{PK_i} is used to encrypt the shares of party P_i, X_i, and Y_i as $E_{PK_{P_i}}(X_i)$ and $E_{PK_{P_i}}(Y_i)$. Because adversary $\mathcal{A}$ lacks knowledge of P_i's private key, it is unable to gain information about X_i and Y_i. As a result, the confidentiality of our proposed protocol is preserved.

***Claim* 2:** The proposed protocol preserves the integrity of the data and it is secure against man-in-the-middle attack.

Proof: Party P_i generates $C_{X_i} \leftarrow E_{PK_{P_i}}(X_i)$, $C_{Y_i} \leftarrow E_{PK_{P_i}}(Y_i)$ and then generates signature $S_i \leftarrow H_{SK_{P_i}}((C_{X_i}, C_{Y_i})||T_i)$. An adversary $\mathcal{A}$ tries to capture the ciphertexts and alter the data and then sends it to another party. However, this is impossible due to the fact that the public key of a party is distributed to the other parties. Any party other than P_i upon receiving the messages C_{X_k}, C_{Y_k}, S_k, first verifies $S_k =? H'_{PK_{P_i}}((C_{X_k}, C_{Y_k})||T_k)$. If they are equal, the process will be further progressed to the next step otherwise not. Therefore, the protocol preserves the integrity of data.

Similarly, we can say that the proposed protocol is secure against man-in-the-middle attack.

***Claim* 3:** The proposed protocol provides security against Impersonation Attack.

Proof: In impersonation attack, $\mathcal{A}$ may try to represent itself as a trusted party P_i and may send its shares $C_{X_\mathcal{A}}$, $C_{Y_\mathcal{A}}$ and signature $S_\mathcal{A}$ to party P_k on behalf of a valid party. However, the adversary becomes unsuccessful as party P_k will verify the signature using P_i's public key. Therefore, the proposed protocol provides security against impersonation attack.

***Claim* 4:** The proposed protocol is secure against replay attack.

Proof: In replay attack, an adversary $\mathcal{A}$ tries to replay a message captured from a past session $Session_1$ in a current session $Sesson_2$. However, this replay attack is impossible due to verification of $T_k - T_i < \Delta t$ condition [Where, T_k, T_i and Δt are the timestamp of message receiver, timestamp of the sender, and a predefined threshold value]. Therefore, the proposed protocol is secure against replay attack.

VI. Performance Analysis

We perform the performance analysis of our proposed protocol in terms of Communication and Computation here.

A. Experimental Setup

For evaluating our proposed protocol, we implement each function using the Crypto++ library [35]. For implementation, we use Intel Core i5 2.4GHz processor.

B. Analysis of Computation and Communication of Our Proposed Protocol

In this subsection, we compare our proposed protocol to the proposed works in [16], [17], [19], [22], [24], [28]. The comparison of total computation time between the works proposed in [16], [17], [19], [22], [24], [28] and our proposed protocol is illustrated in Table II. From Table II, we can see that our proposed protocol requires less time than most of the existing works while preserving the security and privacy of the data of each party. The comparison of the total number of operations between the work proposed in [19] and our proposed protocol is shown in Table III, where the total number of encryption, multiplication, addition, and decryption operations is denoted by T_E, T_M, T_A and T_D, respectively. We compare the total number of operations of our proposed protocol with the work proposed in [19], as they are closely related. In our proposed protocol, there are a total of $n(n-1)$ communications between the parties for every n participant in an ongoing session, as party P_1, P_2, P_3,...,P_n need $n-1$, $n-2$, $n-3$,...,0 number of communications, respectively for sending their encrypted share to other parties. Hence, in this way, $(n-1)n/2$ communication is needed for this procedure. In the same procedure, after element-wise encryption, $(n-1)n/2$ communication is needed for sending the result to other parties. It is shown in Table II that our proposed protocol takes more time than two existing works although it is more efficient than the other five works. However, this consumption of time is justified as this is due to the fact that it ensures the integrity of the passing data.

TABLE II
Comparison of Computation Time

Schemes	Total Computation Time (ms)
Shamir *et al.* [16]	1189.17
Naor *et al.* [17]	101.23
Cock *et al.* [19] (Without TI)	119.76
Cock *et al.* [19] (With TI)	387.33
Berry *et al.* [22]	1231.67
Atallah *et al.* [24]	875.65
Liu *et al.* [28]	1123.45
Our Proposed Protocol	136.54

TABLE III
Comparison of Number of Communications

Work	Cock *et al.* (without TI) [19]	Our Proposed Protocol
T_E	$2(n-1)$	$2(n-1)$
T_M	n^2	n^2
T_A	$2n(n-1)$	$n(n-1)$
T_D	$2n$	$2n$

TABLE IV
Security Comparisons with Some Existing Works

Work	DI	DC	RA	IA	MITM
Shamir [16]	✓	✓	×	×	×
Naor *et al.* [17]	✓	✓	×	×	×
Cock *et al.* [19]	×	✓	×	×	×
Berry *et al.* [22]	×	✓	×	×	×
Atallah *et al.* [24]	×	✓	×	×	×
Liu *et al.* [28]	×	✓	×	×	×
Our Protocol	✓	✓	✓	✓	✓

DI: Preserves Data Integrity, DC: Preserves Data Confidentiality,
RA: Resistant to Replay Attack, IA: Resistant to Impersonation Attack,
MITM: Resistant to Man-in-the-middle attack

C. Security Analysis

The comparison between the proposed work and the existing works in [16], [17], [19], [22], [24], [28] is illustrated in Table IV. × and ✓ denote 'Not-satisfy' and 'Satisfy', respectively. It can be concluded from Table IV that the protocol proposed in [19] is unable to address the most important attacks like man-in-the-middle, replay, impersonation attacks, etc., whereas our proposed protocol is able to address the aforementioned attacks.

VII. Conclusion

The main goal of our proposed protocol is to remove the role of the Trusted Initializer (TI) and to propose a secure n-party computation between n parties. We formally verify our proposed protocol with a well-known verification tool Automated Verification of Internet Security Protocols and Applications (AVISPA). We also carry out the performance analysis of our proposed protocol which shows the effectiveness of our proposed protocol in Secure n-party Computation.

REFERENCES

[1] G.M. Garrido, J. Sedlmeir, O. Uludag, I. S. Alaoui, A. Luckow, and F. Matthes Revealing the Landscape of Privacy-Enhancing Technologies in the Context of Data Markets for the IoT: A Systematic Literature Review. *ArXiv:2107.11905 [Cs]*, 2021.

[2] L. Helminger and C. Rechberger. Multi-Party Computation in the GDPR. In *Privacy Symposium 2022-Data Protection Law International Convergence and Compliance with Innovative Technologies (DPLICIT)* , 2022.

[3] J. I. Choi, K. R. B. Butler. Secure Multiparty Computation and Trusted Hardware: Examining Adoption Challenges and Opportunities. *Security and Communication Networks*, 2019.

[4] A. C. Yao. Protocols for Secure Computations. In *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*, 1982, pp. 160-164.

[5] D. W. Archer, D. Bogdanov, Y. Lindell, L. Kamm, K. Nielsen, J. I. Pagter, N. P Smart, R. N. Wright. From Keys to Databases—Real-World Applications of Secure Multi-Party Computation. *The Computer Journal*, pp. 1749-1771, 2018.

[6] P. S. Naidu, R. Kharat, R. Tekade, P. Mendhe and V. Magade. E-Voting System Using Visual Cryptography and Secure Multi-Party Computation. In *2016 International Conference on Computing Communication Control and Automation (ICCUBEA)*, pp. 1-4, 2016.

[7] D. G. Nair, V. P. Binu, and G. S. Kumar. An Improved E-Voting Scheme using Secret Sharing based Secure Multi-Party Computation. *arXiv*, 2015.

[8] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, L. v. d. Maaten CRYPTEN: Secure Multi-Party Computation Meets Machine Learning. In *35th Conference on Neural Information Processing Systems (NeurIPS 2021)*, 2021.

[9] A. Agarwal, R. Dowsley, N. D. McKinney, D. Wu, C. -T. Lin, M. De Cock, A. C. A. Nascimento. Protecting Privacy of Users in Brain-Computer Interface Applications. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, pp. 1546-1555, 2019.

[10] K. Şahinbaş, and F. O. Catak. Secure Multi-Party Computation based Privacy Preserving Data Analysis in Healthcare IoT Systems. *arXiv*, 2021.

[11] P.K. Tysowski and M.A. Hasan. Re-Encryption-Based Key Management Towards Secure and Scalable Mobile Applications in Clouds. IACR CryptologyEPrint Archive 2011 (2011): 668.

[12] A. López-Alt, E. Tromer, and V. Vaikuntanathan. On-The-Fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing (STOC '12)*, pp. 1219-1234, 2012.

[13] O. Goldreich, S. Micali, and A. Wigderson. How to Play ANY Mental Game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing (STOC '87)*, Association for Computing Machinery, New York, 1987 pp. 218–229.

[14] M. O. Rabin. How To Exchange Secrets with Oblivious Transfer. *IACR Cryptology ePrint Archive*, 2005.

[15] B. Chor, O. Goldreich, E. Kushilevitz and M. Sudan. Private Information Retrieval. In *Proc. of 36th IEEE Conference on the Foundations of Computer Science (FOCS)*, October 1995, pp. 41-50.

[16] A. Shamir. How to Share a Secret. *Commun. ACM 22*, 11 (Nov. 1979), pp. 612–613.

[17] M. Naor and B. Pinkas. Oblivious Transfer and Polynomial Evaluation. In *Proceedings of the thirty-first annual ACM symposium on Theory of Computing (STOC '99)*, Association for Computing Machinery, New York, NY, USA, pp. 245–254.

[18] J. B. Nielsen, P. S. Nordholt, C. Orlandi and S. S. Burra A New Approach to Practical Active-Secure Two-Party Computation. *Advances in Cryptology–CRYPTO 2012*, 2012.

[19] M. D. Cock, R. Dowsley, A.C.A. Nascimento, and S.C. Newman. Fast, Privacy Preserving Linear Regression over Distributed Datasets based on Pre-Distributed Data. *In Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security (AISec '15)*, pp. 3-14, 2015.

[20] P. Pullonen, D. Bogdanov and T. Schneider. The Design and Implementation of a Two-Party Protocol Suite for SHAREMIND 3. CYBERNETICA Institute of Information Security, 2012.

[21] T. Geng, L. Njilla, and C. T. Huang. Delegated Proof of Secret Sharing: A Privacy-Preserving Consensus Protocol Based on Secure Multiparty Computation for IoT Environment. *Network*, pp. 66-80, 2022.

[22] C. Berry and N. Komninos. Efficient Optimisation Framework for Convolutional Neural Networks with Secure Multiparty Computation. *Computers and Security*, 2022.

[23] M. Seo. Fair and Secure Multi-Party Computation with Cheater Detection. *Cryptography*, 2021.

[24] M. Atallah, M. Bykova, J. Li, K. Frikken, and M. Topkara. Private Collaborative Forecasting and Benchmarking. In *Proceedings of the 2004 ACM workshop on Privacy in the Electronic Society (WPES '04)*, Association for Computing Machinery, New York, NY, USA, pp. 103–114.

[25] O. Catrina and S. D. Hoogh. Improved Primitives for Secure Multiparty Integer Computation. *Security and Cryptography for Networks*, pp. 182-199, 2012.

[26] O. Catrina and A. Saxena. Secure Computation with Fixed-Point Numbers. *Financial Cryptography and Data Security*, pp. 35-50, 2010.

[27] D. Beaver. One-Time Tables for Two-Party Computation. *Computing and Combinatorics*, pp 361-370, 1998.

[28] Y. Liu, Z. Ma, X. Liu, S. Ma and K. Ren. Privacy-Preserving Object Detection for Medical Images With Faster R-CNN. *IEEE Transactions on Information Forensics and Security*, pp. 69-84, 2022.

[29] L. Li and C. Chen. Secure Multi-Party Computation with No Trusted Initializer. U.S. Patent Application No. 16/668,945. October 8, 2020.

[30] Y. Jiang, K. Zhang, Y. Qian and L. Zhou. Anonymous and Efficient Authentication Scheme for Privacy-Preserving Distributed Learning. *IEEE Transactions on Information Forensics and Security*, pp. 2227-2240, 2022.

[31] R. L. Rivest. Unconditionally Secure Commitment and Oblivious Transfer Schemes Using Private Channels and a Trusted Initializer. Preprint available at http://people.csail.mit.edu/rivest/Rivestcommitment.pdf, 1999.

[32] T. Sridokmai and S. Prakancharoen. The Homomorphic Other Property of Paillier Cryptosystem. In *Proceedings of 2015 International Conference on Science and Technology (TICST)*, pp. 356-359, 2015.

[33] D. Dolev and A. Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198-208, March 1983.

[34] J. Srinivas, A. K. Das, N. Kumar, and J. J. P. C. Rodrigues. Cloud Centric Authentication for Wearable Healthcare Monitoring System. *IEEE Transactions on Dependable and Secure Computing*, 17(5):942–956, 2020.

[35] Crypto++ Library. https://cryptopp.com/