electronics

MDPI

Article

Efficient Decision-Making Scheme Using Secure Multiparty Computation with Correctness Validation

Tao Wang [1], Zhusen Liu [2], Zhaoyang Han [1] and Lu Zhou [1,*]

[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210000, China; tao.wang@nuaa.edu.cn (T.W.); sunrisehan@nuaa.edu.cn (Z.H.)
[2] Research Center for Basic Theories of Intelligent Computing, Research Institute of Basic Theories, Zhejiang Lab, Yuhang District, Hangzhou 310000, China; liuzs@zhejianglab.com
* Correspondence: lu.zhou@nuaa.edu.cn

Abstract: In the era of big data, it is essential to securely and efficiently combine the large amounts of private data owned by different companies or organizations to make correct decisions. Secure Multiparty Computation (SMPC) works as a general cryptographic primitive, which enables distributed parties to collaboratively compute an arbitrary functionality without revealing their own private inputs. While SMPC may potentially address this task, several issues, such as computation efficiency and correctness validation, have to be overcome for practical realizations. To tackle these issues, we designed a secure and efficient decision-making scheme to enable clients to outsource data and computations to cloud servers while ensuring the integrity and confidentiality of the input and output, in addition to the correctness of the results. Moreover, we implemented our scheme based on an SMPC computation framework named MP-SPDZ. The experimental evaluation results showed that our proposed scheme is feasible and efficient for practical realizations.

Keywords: secure multiparty computation; health insurance; decision-making; correctness validation

Citation: Wang, T.; Liu, Z.; Han, Z.; Zhou, L. Efficient Decision-Making Scheme Using Secure Multiparty Computation with Correctness Validation. *Electronics* **2023**, *12*, 4840. https://doi.org/10.3390/electronics12234840

Academic Editors: Rashid Mehmood and Andrei Kelarev

Received: 18 October 2023
Revised: 25 November 2023
Accepted: 27 November 2023
Published: 30 November 2023

1. Introduction

With the flourishing development of the big data era, data are growing at an exponential rate every year, especially private data in business environments. Most companies and organizations that possess private data, such as healthcare centers, electronic commerce companies, online banks, and insurance companies, are unwilling to trade data in plaintext form. In today's distributed network environment, which uses vulnerable and diverse communication channels, performing various computations on data in plaintext form inevitably comes with the issues of private data or confidential information leakage caused by network deception attacks or eavesdropping [1]. Therefore, a solution to combine data owned by different companies or organizations to make correct decisions while also protecting data privacy is urgently needed and will greatly increase the value of the data.

In general, traditional privacy-preserving techniques used in industry mainly include data anonymization [2], anonymity algorithms [3], differential privacy [4], and data encryption. These different privacy-preserving techniques protect privacy by changing sensitive data to conceal or erase the original sensitive information. Our goal was to find a solution that can combine large amounts of sensitive data from different companies or organizations while ensuring the confidentiality of the data. Unfortunately, data anonymization and anonymity algorithms may lose their effectiveness in protecting data privacy under the powerful data analysis capabilities of big data. For example, by conducting correlation analyses and carrying out deep mining on multi-source data, users' privacy information can be easily restored. Differential privacy achieves privacy preservation by transforming the original data or adding noise to the statistical results, but for some complex queries, the addition of noise may lead to excessive errors. Data encryption utilizes cryptographic

techniques to transform sensitive data into ciphertext, attempting to not reveal any information about the original data unless it can be inferred from the task output. Secure Multiparty Computation (SMPC) is a typical cryptography-based method. In recent years, SMPC has been widely applied in real-world applications to achieve privacy protection, such as disease diagnosis [5], bank customer risk prediction [6], etc., because it is more effective and efficient than other methods. SMPC enables distributed parties to collaboratively compute any function without revealing their own private inputs and outputs. For large-scale data in real-world scenarios, it can also support efficient fixed-point and floating-point operations [7], and arithmetic operations can be achieved with controlled linear complexity [8]. Due to these advantages, researchers have shifted their focus from purely theoretical research to real-world applications. However, there are still some problems, such as efficiency, computational overhead, and validation, that need to be addressed in practical applications.

Traditional SMPC is often unable to address the issues of low efficiency and unverifiable results when it comes to solving privacy-preserving computations across institutions. In this paper, we considered a novel real-world scenario, i.e., health insurance, to bridge the medical field with the financial sector's data to achieve secure and efficient privacy-preserving health insurance decision-making. To support these requirements, we utilized the SPDZ protocol [9,10] hypothetically under a malicious security model. Meanwhile, in a realistic situation, insurance companies typically need a third-party institution for notarization or verification when making health insurance decisions. Towards this end, we extended the SPDZ protocol by adding an Honest Validator (HV) as a third party to verify the results. Doing so increases the computation and communication costs of SMPC nodes, but a slight loss of efficiency is deemed tolerable for the real application. We implemented the proposed protocol using the open-source MP-SPDZ framework [11] and safely calculated the decision-making results with it. To further accommodate potential real-world applications, we designed programs that can support parallel computing to improve the efficiency. In summary, our goal in this work was to design a scheme that combines data owned by various companies or organizations to make correct decisions while also safeguarding data privacy.

The primary contributions of this paper are as follows:

- For the health insurance scenario, based on data vectorization, Pedersen commitments, and non-interactive zero-knowledge proofs, we designed and implemented an efficient scheme to ensure the security of private data. To the best of our knowledge, this is the first work that applies Secure Multiparty Computation to the health insurance scenario.
- We propose a specific Secure Multiparty Computation protocol based on our scheme. In detail, we extended the SPDZ protocol by adding a third party, i.e., an Honest Validator (HV), to verify the correctness of the result.
- We implemented the proposed scheme on the open-source MP-SPDZ framework and carried out experiments with a large amount of real-world data. The key performance metrics, transmission data size, bytecode size, and execution time, demonstrated the feasibility and effectiveness of our model for health insurance decision-making scenarios. Furthermore, compared to traditional approaches, where any computing party can act as a validator, our scheme demonstrated approximately 10% improved efficiency.

The rest of this paper is organized as follows. In Section 2, we briefly introduce related works on SMPC and its applications in the medical field and the financial sector. Then, in Sections 3 and 4, we introduce our system framework, threat model, and some fundamental knowledge of our scheme. The proposed scheme for the health insurance scenario is discussed in Section 5. In Section 6, we present the experimental results of our scheme to show its effectiveness. Finally, we summarize this work and discuss our future work.

2. Related Work

Collecting data from multiple parties to make correct decisions is an important task. Secure Multiparty Computation works as a general cryptographic primitive, which enables

joint computing in a privacy-preserving manner. Yao [12] first proposed the problem of secure two-party computation and its solution in 1982. Then, researchers extended secure two-party computation to the case of multiple parties [13,14] and developed many high-performance multiparty protocols with the help of techniques such as oblivious transfer, secret sharing, zero-knowledge proofs, garbled circuits, and homomorphic encryption. The SPDZ (pronounced "speedz") protocol [10] was one of them. It emerged as one of the most-effective and practical because it shifted the vast majority of computation to an offline phase (also known as preprocessing phase), which can be performed locally in advance. In the following decade, researchers have proposed several variants of the protocols [15–17] based on the SPDZ protocol further to improve its computation and communication efficiency, while also paying attention to specific applications.

There have been many works to investigate the application of SMPC in the medical field or financial sector. The study in [18] demonstrated that Secure Multiparty Computation using garbled circuits is a viable technology for addressing clinical use cases that require cross-institutional data exchange and collaboration. Li et al. [5] proposed a secure and efficient assisted multiparty decision-making scheme appropriate for Internet of Medical Things applications. Garofalakis [19] leveraged SMPC to provide an end-to-end infrastructure for computing privacy-preserving analytics on confidential healthcare data. The authors in [6,20] analyzed private data in the financial sector such as banks, insurance, and securities using SMPC. However, we noticed that most of the work using SMPC applications has not yet addressed the implementation in cross-institutional scenarios. In this paper, we overcame the obstacles and united different fields soundly to perform secure computation.

Concerning the validability of the results, to the best of our knowledge, this was first employed in electronic voting systems [21] and was termed public verifiability. Later, widely known publicly auditable voting protocols such as [22,23] were proposed. Among these voting systems, most use commitment mechanisms and zero-knowledge proofs to enforce verifiability. Baum et al. [24] first integrated Pedersen commitments [25] with an efficient information-theoretic SPDZ protocol [10] and proposed a protocol for which any party could function as an auditor, with approximately twice the efficiency of the original SPDZ protocol. Then, the work in [26–29] built on this foundation and proposed protocols with higher security requirements or efficiency. Our work utilized a single-party HV as the validator, which has the merit that the validation phase can be executed simultaneously with the computational protocol.

3. System Model

In this section, we introduce a detailed model of the health insurance scenario system, including the roles of all participants and their functions.

3.1. System Model

The scheme has four distinct types of participants, as shown in Figure 1.

Users: When purchasing health insurance, visiting a hospital when sick, and the reimbursement of the costs occurs, the user's personal information, including sensitive data such as occupation and annual income, as well as the records of the medical visits, are stored in the relevant institutions.

Healthcare institutions: These include healthcare centers and insurance companies. They each own a large amount of information about users and desire to leverage both parties' resources to make health insurance decision-making for financial benefit. Before that, they need to perform unified preprocessing of their respective data according to trusted third parties' standards.

Cloud providers: They are also referred to as SMPC nodes. They receive the data from healthcare institutions and perform secure calculations using SMPC while transmitting the parameters required for subsequent verification to the validator.

Honest validator: This is also a cloud provider, unless it validates the result from the SMPC nodes, thus guaranteeing the computation's correctness.

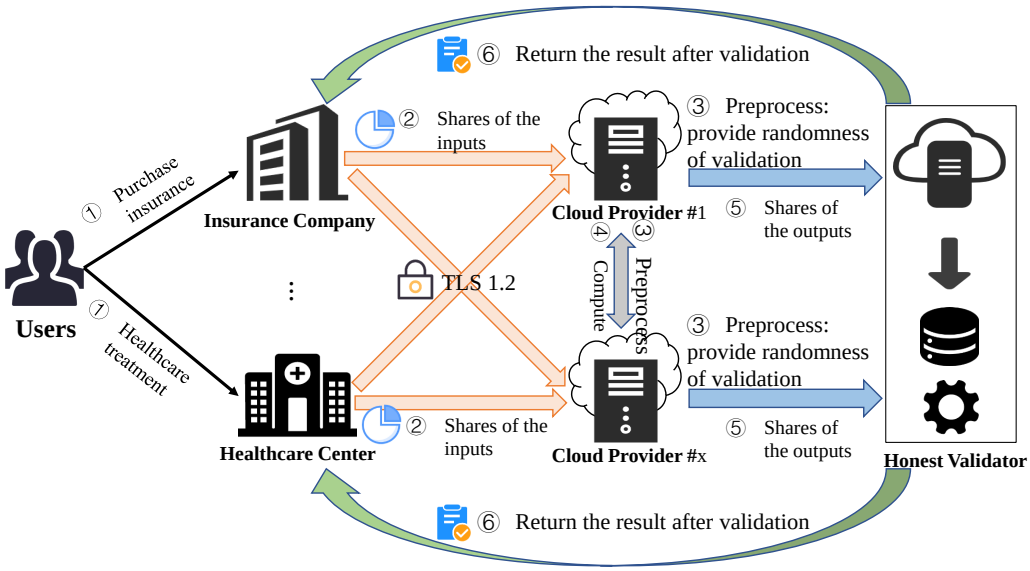

Figure 1. System scheme with Secure Multiparty Computation.

3.2. Threat Model

As depicted in Figure 1, the scheme consists of four participants, including one honest validator, which can be relied upon for its credibility. The remaining three participants, however, may not be trustworthy. In this scenario, we assumed the presence of an adversary, denoted as $\mathcal{A}$, who may gain access to data outside of the honest validator.

$\mathcal{A}$ may launch an attack on the cloud to obtain encrypted data stored by users there. Similarly, $\mathcal{A}$ can obtain encrypted data from all parties by eavesdropping the network communication link. The worst-case scenario is that $\mathcal{A}$ may be able to collaborate with other participants to deduce users' private information from the ciphertext. Security and efficiency will be realized through the proposed protocol to address these concerns in this system.

4. Preliminaries

In this section, we introduce some crucial notations, define the secret-sharing primitive, and give a basic overview of the SPDZ protocol, the Pedersen commitment, and non-interactive zero-knowledge proofs.

Notions: The protocol we present for the health insurance scenario in this paper considers arithmetic circuits over p-order fields, where p is a large Sophie Germain prime, that is $q = 2p + 1$ is also a prime. $\mathbb{Z}_p$ refers to the field $\{0, \ldots, p-1\}$; $\mathbb{Z}_p^*$ refers to $\mathbb{Z}_p \backslash \{0\} = \{1, \ldots, p-1\}$. We claim that an element g of a cyclic group G is a generator of this group only if $\forall x \in G$, $\exists a$ satisfies $g^a = x$. We write $\langle x \rangle_i$ to denote the share of value x in the original SPDZ protocols and use $[x]_i$ to denote the share of the validation phase x.

We use $\mathcal{P}_i \in \mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ to identify a compute party (i.e., a server in the client–server model) and use $\mathcal{P}_i^I \in \mathcal{P}^I = \{\mathcal{P}_1^I, \ldots, \mathcal{P}_m^I\}$ to identify an input party (i.e., a client). We use $\mathcal{C} \subseteq \mathcal{P}$ for the set of statically corrupted parties and $\mathcal{H} = \mathcal{P} \setminus \mathcal{C}$ for the setting of honest parties. The input side $\mathcal{P}_i^I$ can also be statically corrupted, but all our results are independent of the exact set of corrupted input sides $\mathcal{C}$.

4.1. The SPDZ Protocol

SDPZ is a state-of-the-art SMPC protocol, which operates on secret-sharing principles, enabling joint computation of arithmetic circuits. It allows shareholders to compute linear functions of secret inputs while ensuring that they learn nothing beyond the final output of

those functions. Moreover, SPDZ secret sharing exhibits homomorphic properties, which means that addition operations can be performed on the shares without revealing the underlying values. To illustrate this, suppose X and Y are shared as x_1, x_2 and y_1, y_2, respectively, where $X = \sum_i x_i \bmod p$ and $Y = \sum_i y_i \bmod p$ for some prime p. In that case, $(x_1 + x_2)$ and $(y_1 + y_2)$ will also be shares of $(X + Y)$. The SPDZ protocol operates in two phases: an offline preprocessing phase, which generates the necessary raw materials, and an online evaluation phase, which executes the circuit.

Offline phase: This utilizes the Somewhat Homomorphic Encryption (SHE) [30] scheme to generate an ample supply of triples that can be used in multiplication operations based on Beaver's method [31]. For instance, to compute $\langle xy \rangle$, given some $\langle x \rangle$ and $\langle y \rangle$, it is necessary to possess three secret-sharing values called a triple: $\langle a \rangle$, $\langle b \rangle$, and $\langle c \rangle$, where $c = ab$. It is worth noting that this phase can generate as many triples as required in advance, independent of the input data and function to be computed.

Besides, in the primitive SPDZ protocol, a global private key denoted by α is generated during the offline phase. This key serves as a Message Authentication Code (MAC) and is utilized by the participating nodes to verify the "correctness" of the computation results. Notably, α is distributed secretly among the nodes, such that $\alpha = \sum_i \alpha_i$ and each node possesses α_i. Unfortunately, despite its efficacy in guaranteeing the integrity and authenticity of transmitted messages, MACs alone cannot verify the results' correctness.

Online phase: The online phase evaluates the function. Some operations on shares, like addition or multiplication by a constant, can be performed without communication among computation nodes. Then, we define a share of x as

$$\langle x \rangle = \{(x_1, \ldots, x_n), (\delta(x)_1, \ldots, \delta(x)_n)\}$$

where $x = \sum_{i=1}^{n} x_i$ and $\alpha \cdot x = \sum_{i=1}^{n} \delta(x)_i$. Moreover, we define

$$\begin{gathered} \langle x \rangle + \langle y \rangle = \{(x_1 + y_1, \ldots, x_n + y_n), (\delta(x)_1 + \delta(y)_1, \ldots, \delta(x)_n + \delta(y)_n)\} \\ \langle x \rangle \cdot s = \{(x_1 \cdot s, \ldots, x_n \cdot s), (\delta(x)_1 \cdot s, \ldots, \delta(x)_n \cdot s)\} \\ \langle x \rangle + s = \{(x_1 + s, x_2, \ldots, x_n), (\delta(x)_1 + \alpha_1 \cdot s, \ldots, \delta(x)_n + \alpha_n \cdot s)\} \end{gathered}$$

where $y = \sum_{i=1}^{n} y_i$ and s is a constant. When opening a shared value $\langle x \rangle$, parties first broadcast their shares x_i and compute x. To ensure that x is correct, they then check the MACs by committing to and opening $\delta(x)_i - x \cdot \alpha_i$ and checking that these shares sum up to zero.

4.2. The Pedersen Commitment

Considering the linear nature of the existing SPDZ framework, our proposal entails leveraging a Pedersen commitment [25] scheme as a means of validating the correctness of the computation results. Let g and h be elements of G such that nobody knows $\log_g h$. When the protocol is initialized, these elements should be chosen in the offline phase. When the committer commits himself/herself to an $x \in \mathbb{Z}_p$, the commitment function chooses $r \in \mathbb{Z}_p$ at random and computes

$$\mathcal{W}(x, r) = g^x h^r \in G.$$

Such a commitment can be later revealed by x and r. It is straightforward to prove that $\mathcal{W}(x, r)$ indicates no information about x and that the committer opens a commitment to x as $x' \neq x$ unless he/she can find $\log_g h$. Thus, we can say the Pedersen commitment scheme is hiding and binding.

4.3. Non-Interactive Zero-Knowledge Proofs

To achieve the audibility property in the offline phase of our proposed protocol, we utilized efficient Non-Interactive Zero-Knowledge Proofs (NIZKPs) [32]. This is a two-party protocol between a prover and a verifier, where the prover convinces the verifier that a statement is accurate and the verifier learns nothing beyond the claim statement. It can be expressed as follows:

(1) The prover chooses $t_1, t_2 \in \mathbb{Z}_n$, then computes $T = g^{t_1} h^{t_2}$ and sends T to the verifier.
(2) The verifier chooses $c \in \mathbb{Z}_n$ and sends c to the prover.
(3) After receiving c, the prover calculates $s_1 = t_1 + cx \bmod \phi(n)$ and $s_2 = t_2 + cr \bmod \phi(n)$ and sends s_1, s_2 to the verifier.
(4) Finally, the verifier checks $Tc^c = g^{s_1} h^{s_2} \bmod \phi(n)$.

The prover proves knowledge of an element x and decommitment value r such that $Com = g^x h^r \bmod n$.

5. Our Scheme

In this section, we will provide a detailed description of our scheme.

5.1. Overview

As shown in Figure 1, we designed a novel and efficient scheme to combine private information about users in healthcare centers and insurance companies to make valuable decisions while validating the correctness of decision-making results. The scheme is based on the SPDZ protocol and Pedersen commitments and is divided into offline and online phases. The offline phase primarily provides the necessary preprocessing and information exchange for the subsequent online computation phase and generates the required parameters for validation. The online phase enables the participants to securely compute the decision results together, without revealing their respective private inputs and ultimately returns the results to HV for correctness validation. The decision-making algorithm used in the online computation phase is derived from real-world applications, which will be described in detail in the last subsection of this section.

Online phase: In this phase, SMPC nodes carry out the computational steps and obtain the required decision-making results while protecting privacy. However, to validate the results, modifying the shared values in the original SPDZ protocol with Pedersen commitments is essential. On the other hand, we assumed that there is a transcript ϵ on the HV to store the communication data sent by the computational nodes, and we used Algorithm 1 to keep an eye on all values needed for validation.

Algorithm 1: Store_Load—algorithm for storing and loading message on the transcript ϵ.

Input:
$(tag, i, msg) \rightarrow$ The communication data sent by SMPC nodes
$N \in \mathbb{Z}^+ \rightarrow$ The number of SMPC nodes
Output:
(tag, i, msg) or $(\dashv) \rightarrow$ The data to be validated or the error flag

```
// store message on the transcript ϵ
for j = 0 to N − 1 do
    store (tag, i, msg) on ϵ;
end
// load message on the transcript ϵ
foreach load_msg do
    if (tag, i) on ϵ then
        load (msg) to HV for validation;
    else
        return ⊣;
    end
end
```

Offline phase: As the name suggests, this phase can be completed in advance, allowing it to generate the necessary randomness for the online computation phase. For example, it can createsecret keys for message authentication codes, ample multiplication triples,

commitment parameters required for result validation, etc. These preprocessing and generation of necessary parameters in the offline phase significantly enhance the efficiency of online computation. The SMPC nodes run the offline phase of the protocol, preparing all correlated randomness for the online phase, including some parameters necessary for the validator.

5.2. Online Phase

To enforce the validation of the output, it is essential to modify the shared values in the original SPDZ protocol. Specifically, we propose altering the inputs, opened values, and outputs of the computational nodes at the point of their commitment. These modified values are then transmitted to the transcript ε on the HV, where the correctness of all intermediate steps can be verified. The Pedersen commitment scheme is information-theoretically hiding during the whole computation process. When we open a Pedersen commitment $\mathcal{W}(x, x_r)$, we reconstruct both x and x_r, and the HV can check that it is correct.

Let $x, x_r \in \mathbb{F}$, $g, h \in G$, and then, we define the new way to represent shared value x with a Pedersen commitment as

$$[x] = \{\langle x \rangle, \langle x_r \rangle, \mathcal{W}(x, x_r)\},$$

where $\langle x \rangle$ is a representation of x as introduced in Section 4.1. Similarly, with this new linear representation, we can perform the same operations linearly as in Section 4.1. We can still employ Beaver's circuit randomization technique for the multiplication of $[x]$ and $[y]$ since the representation is linear.

Given that operations in the online phase depend on multiplication triples and the randomness required for the validation phase, we define the functionality $\mathcal{F}_{offline}$ that describes the behavior and output of the preprocessing. In general, $\mathcal{F}_{offline}$ must perform the following: (1) establish an additively secret-sharing MAC key α; (2) generate random $[x_r]$ shared values to be used during **Input**; (3) generate $[a, b, c]$ shared Beaver triples for every multiplication to be performed in the online phase.

In Figure 2, we present the protocol of the online phase. It uses $\mathcal{F}_{offline}$ for the offline phase and the transcript ϵ for all communication.

At the beginning step **Prepare**, the parties $\mathcal{P}$ generate the parameters g, h of the Pedersen commitment via a Common Reference String (CRS), establish the secret shared MAC key α, as well as produce views for random triples a, b, c, where $c = a \cdot b$. Once the offline phase is called to obtain the initial randomness, a new input-independent view $[r]_i$ can be utilized to generate the values for the **Input** step. Each party $\mathcal{P}_i^I \in \mathcal{P}^I$ submits a value $x_i \in \mathbb{F}$ to the computation, where each $\mathcal{P}_i^I$ uses the new $[r]_i$ to secretly open it. Then, each can check the correctness of the commitment on it and use $m_i = x_i - r_i$ to mask his/her input.

Similarly, in **Compute**, the provided triples ($[a]$, $[b]$, $[c]$) will be used in the computation. They are used to multiply with linear operations as follows:

$$[x \cdot y] = [c] + \sigma \cdot [b] + \tau \cdot [a] + \sigma \cdot \tau.$$

$[\sigma] = [x] - [a]$ and $[\tau] = [y] - [b]$ are opened values. Other linear operations (addition, subtraction, or with publicly known constants) can be performed locally. The last operation left for the step is outputting the results. Here, we ran the MAC check to verify the result y and its correlated $\overline{y}$; thus, the parties can open the output and send the result to transcript ε pending further validation.

In the final step **Validate**, the validator HV will execute the calculation process gate by gate as was performed by $\mathcal{P}_i$. For the records on the transcript ε, the HV checks every opened value. When this is performed, the result of the computation and the commitment of [y] will finally be checked for correctness against the views on ε.

Π_{Online}

Assume that each party $\mathcal{P}_i$ can obtain validated randomness and triples from $\mathcal{F}_{offline}$.

1. **Prepare**: Parties choose commitment parameters $g, h \in \mathbb{G}$ at random via Common Reference String. Then send them to $\mathcal{F}_{offline}$, and get the shares α_i, sufficient Input $[r]$ and Triples $[a], [b], [c]$. If $\mathcal{F}_{offline}$ return $\dashv$, abort.
2. **Input**: Each party $\mathcal{P}_i^I \in \mathcal{P}^I$ inputs a value $x_i \in \mathbb{F}$, and then each $\mathcal{P}_i^I \in \mathcal{P}^I$, $\mathcal{P}_i \in \mathcal{P}$ do the same operations:
   - Let $[r]_i$ be a new random value, and it is opened as r_i,$\bar{r}_i$ only to $\mathcal{P}_i^I$. Com_{r_i} is the commitment of $[r]_i$ on the transcript ε. $\mathcal{P}_i^I$ check that $Com_{r_i} = \mathcal{W}(r_i,\bar{r}_i)$. If not, abort.
   - Then $\mathcal{P}_i^I \in \mathcal{P}^I$ computes $m_i = x_i - r_i$ and sends to all $\mathcal{P}_i$ and the ε.
   - All $\mathcal{P}_i \in \mathcal{P}$ compute $x_i = r_i + m_i$ locally.
3. **Compute**: There are two basic operations in the online phase, and other operations can be varied from these two ones.

   a. Add: Given two values $[x], [y]$ with corresponding tags tag_x, tag_y.
   - Each party $\mathcal{P}_i \in \mathcal{P}$ computes $[z] = [x] + [y]$ locally, and assigns a new tag tag_z to it.

   b. *Multi*: Let $[x]$ and $[y]$ are multiplied with tags tag_x, tag_y. It is inevitable to use the multiplication triple $([a], [b], [c])$ where $c = a \cdot b$.
   - All parties compute $[\sigma] = [x] - [a]$ and $[\tau] = [y] - [b]$.
   - Then they reconstruct $\sigma, \bar{\sigma}, \tau, \bar{\tau}$ in public and send these values to ε.
   - Each $\mathcal{P}_i \in \mathcal{P}$ locally computes $[z] = [c] + \sigma \cdot [b] + \tau \cdot [a] + \sigma \cdot \tau$, and assigns a new tag tag_z to it.

   c. *Result*: All parties open the output $[y]$ (maybe $[y]$ is the output in the case of addition, multiplication or other combinations).
   - Parties do MACs check, compute $\alpha \cdot y - \delta(t) = 0$ and $\alpha \cdot \bar{y} - \delta(\bar{y}) = 0$. If not, abort.
   - Then parties open the output $[y]$ and send it to ε.
4. **Validate**: For each gate in the calculation process, HV do the following operations:
   - The HV adds $[x]$ (with tag tag_x) and $[y]$ (with tag tag_y) to $[z]$ (with a new tag tag_z). Set $Com_{tag_z} = Com_{tag_x} \cdot Com_{tag_y}$.
   - The HV multiplies $[x]$ and $[y]$ to $[z]$ with sufficient random values $[a], [b], [c], [\sigma], [\tau]$ (with corresponding respective tag). Set
   $$Com_{tag_z} = Com_{tag_c} \cdot Com^{\sigma}_{tag_b} \cdot Com^{\tau}_{tag_a} \cdot \mathcal{W}(\sigma \cdot \tau, 0),$$
   and then check $Com_{tag_x} = Com_{tag_a} \cdot \mathcal{W}(\sigma, \bar{\sigma})$ and $Com_{tag_y} = Com_{tag_b} \cdot \mathcal{W}(\tau, \bar{\tau})$. If not, abort.
   - Let y be the output and Com_y be the commitment for the output value $[y]$. If $Com_y = \mathcal{W}(y, \bar{y})$, return y to all $\mathcal{P}_i^I$. Otherwise, return $\dashv$.

Figure 2. Protocol Π_{Online}.

5.3. Offline Phase

As described above, the offline phase has to produce views of secure MAC key α, views of Beaver triples, and a masked input m.

We reused a homomorphic encryption scheme like Damgård et al. in [10] to achieve these goals. Let $\mathcal{S} = (KeyGen, Enc, Dec)$ be such a scheme. It must support the evaluation of circuits with polynomially many additions and one homomorphic multiplication. It should also facilitate distributed vital generation. Each participant obtains a share of the secret decryption key in this process, while all parties possess the public encryption key. Then, the secret-sharing decryption key will be utilized for distributed decryption, which involves processing a ciphertext and distributing the resulting plaintext to each party involved. Here, each participant can employ NIZKPs to guarantee the correctness and security of scheme $\mathcal{S}$.

The first step is to generate a secret-sharing MAC key α, which is used to calculate the product with the input shared value and, then, Reshare the result. As shown in Algorithm 2, it is a functionality for distributed MAC key generation. Its result is a ciphertext, and we can learn the value by using distributed decryption in $\mathcal{S}$. However, we really need a $[x]$-shared

value. A sub-protocol $\Pi_{Reshare}$ (shown in Algorithm 3) can produce a ciphertext as a plain $[x]$-shared value.

Algorithm 2: MacKeyGen—algorithm for generating a secret-sharing MAC key.

Input:
$(pk, \langle sk \rangle_i) \rightarrow$ The public encryption key and the sharing of the decryption key for the SHE scheme $\mathcal{S}$
$N \in \mathbb{Z}^+ \rightarrow$ The number of SMPC nodes
Output:
$\alpha \in \mathbb{Z}_p \rightarrow$ The secret MAC key

for $j = 0$ **to** $N - 1$ **do**
  // randomly choose a number
  $\alpha_i = random(\mathbb{Z}_p)$;
  // compute the ciphertext along with an NIZKPs
  $e_{\alpha_i} = Enc(pk, \alpha_i)$ *and* $c_{e_{\alpha_i}} = NIZKPs(e_{\alpha_i})$;
  send $(e_{\alpha_i}, c_{e_{\alpha_i}})$ *to all other parties and the transcript* ϵ;
end
when receiving ciphertexts and the corresponding NIZKPs from all other parties
foreach $(e_{\alpha_i}, c_{e_{\alpha_i}})$ **do**
  if $c_{e_{\alpha_i}}^{-1} = e_{\alpha_i}$ **then**
    // use additive homomorphic properties of the scheme $\mathcal{S}$
    $e_\alpha = \sum_{i=0}^{N-1} e_{\alpha_i}$;
  else
    return $\dashv$;
  end
end

With the algorithm *Reshare*, similarly, we can easily generate a $[x]$-shared random value. Now, the rest of the preprocessing phase is how to generate Beaver triples. Like the above functionality, we assumed that each party $\mathcal{P}_i \in \mathcal{P}$ already knows the public encryption key pk and has an additive secret sharing of the decryption key sk. Simultaneously, parameters (g, h) for the Pedersen commitments are known. The parties are assumed to have run the algorithm *MacKeyGen*. Thus, each party has an additive secret sharing of the MAC key α and an encryption e_α. Now, we briefly describe the functionality *GenTriples*:

(1) Using the same method in the algorithm *Reshare*, each party $\mathcal{P}_i$ can hold two random $[x]$-shared values $[a]$ and $[b]$. Each party calculates ciphertexts $e_{a_i} = Enc(pk, \langle a \rangle_i)$ and $e_{b_i} = Enc(pk, \langle b \rangle_i)$ together with the NIZKPs of both. Then, they broadcast e_{a_i}, e_{b_i}, and the NIZKPs to all other parties and ϵ. When each receives the ciphertexts and the corresponding NIZKPs from all parties, check all the NIZKPs. If cheaters exist, abort.
(2) Utilizing the additive and multiplicative homomorphic properties of the scheme $\mathcal{S}$, all parties compute: (1) a ciphertext e_a such that $a = \sum_{i=1}^{n} a_i$; (2) a ciphertext e_b such that $b = \sum_{i=1}^{n} b_i$; (3) an encryption $e_{a \cdot b}$ of the product of a and b.
(3) The parties run the algorithm *Reshare* on $e_{a \cdot b}$, resulting in a secret sharing $[c]$, where $c = a \cdot b$.

Finally, we can assemble these components to obtain the offline protocol $\Pi_{Offline}$ (shown in Figure 3). Steps 1–4 utilize the algorithms described above to generate the needed randomness for the online computation. Likewise, these generated random values need to be validated. Benefiting from the fact that the corresponding commitments and NIZKPs were generated at each phase and sent to ϵ, the HV can validate their correctness in **Validate**.

Algorithm 3: Reshare—algorithm for producing a plain $[x]$-shared value.

Input:
$(pk, \langle sk \rangle_i) \rightarrow$ The public encryption key and a sharing of the decryption key for the SHE scheme $\mathcal{S}$
$(g, h) \rightarrow$ The generators of Pedersen commitments
$(\alpha_i, e_{\alpha_i}) \rightarrow$ A share of the MAC key and the corresponding encryption e_α
$N \in \mathbb{Z}^+ \rightarrow$ The number of SMPC nodes
Output:
$[x]_i = \{\langle x \rangle_i, \langle x_r \rangle_i, (com_1, \ldots, com_N)\} \rightarrow$ Each party holds an $[x]$-shared value

for $j = 0$ **to** $N - 1$ **do**
  $f_i, \overline{f_i} = random(\mathbb{Z}_p)$;
  $e_{f_i} = Enc(pk, f_i)$, $com_{f_i} = \mathcal{W}(f_i, \overline{f_i})$ *and* $c_{f_i} = NIZKPs(f_i)$;
  send $(e_{f_i}, com_{f_i}, c_{e_{f_i}})$ *to all other parties and the transcript* ϵ;
end
when receiving the ciphertexts and the corresponding NIZKPs from all other parties
foreach $(e_{f_i}, c_{e_{f_i}})$ **do**
  if $c_{e_{f_i}}^{-1} = e_{f_i}$ **then**
    $e_f = \sum_{i=0}^{N-1} e_{f_i}, e_{x+f} = \sum_{i=0}^{N-1} e_{(x+f)_i}$;
  else
    return $\dashv$;
  end
end
to decrypt e_{x+f}*, all parties learn the value* $x + f$
$r_{x+f} = random(\mathbb{Z}_p), com_{x+f} = \mathcal{W}(x + f, r_{x+f})$;
$\mathcal{P}_0$ *sets* $\langle x \rangle_0 = x + f - x_0, Dec_{\langle x \rangle_0} = Dec_{x+f} - Dec_0$;
$com_0 = com_{x+f} \cdot com_{f_0}^{-1}$;
for $j = 1$ **to** $N - 1$ **do**
  $\langle x \rangle_j = -f_j, Dec_{\langle x \rangle_j} = -Dec_j$;
  $com_j = com_{f_j}^{-1}$;
end
for $j = 0$ **to** $N - 1$ **do**
  // use the multiplicative homomorphic properties of $\mathcal{S}$
  $e_{\delta_x} = \sum_{i=0}^{N-1} e_{\alpha_i} \cdot x$;
end

$\Pi_{Offline}$

Assume that generators g, h for the Pedersen commitment scheme are known.

1. **Prepare**: The parties run distributed key generation of scheme $\mathcal{S}$ to obtain a shared encryption key pair (pk, sk), where pk is public available and sk is an additively secret-shared key among all parties.
2. **GenMacKey**: The parties run $\Pi_{MacKeyGen}$ to obtain a secret-shared MACs key α along with its ciphertext e_α.
3. **Input**: The parties run $\Pi_{Reshare}$ to generate a random $[x]$-shared value for each input.
4. **Triples**: The parties run $\Pi_{GenTriples}$ to generate $[x]$-shares of Beaver triples for each multiplication.
5. **Validate**: For every encryption e, commitment com and corresponding NIZKPs on the transcript ε, HV checks the correctness of NIZKPs and verifies the validity between com and $\mathcal{W}(x, \bar{x})$. If an error message exists, return $\dashv$.

Figure 3. Protocol $\Pi_{Offline}$.

5.4. Decision-Making Algorithm

Based on the protocol above, Algorithm 4 is proposed to describe the decision-making algorithm, which is a case in the health insurance scenario. We used it as the basis of the experimental phase.

Algorithm 4: Dec_Make—algorithm for making a health insurance decision using SMPC.

Input:
$M = [m_{ij}] \in \mathbb{R}^{r\times 4} \rightarrow$ The medical records' matrix with size $r \times 4$
$N = [n_{ij}] \in \mathbb{R}^{r\times 8} \rightarrow$ The insurance information matrix with size $r \times 8$
Output:
$A = [a_{ij}] \in \mathbb{R}^{n\times 9} \rightarrow$ The insurance decision outcome matrix with size $n \times 9$

initialize a matrix $T = [t_{ij}] \in \mathbb{R}^{r\times 9}$ *with 0s*;
for *each row* $m_i, n_i (1 \leq i \leq n) \in M, N$ **do**
  // first identify some specific diseases
  while $spe\_dis == true$ **do**
    $t_{i\times 1} = 1$;
  end
  // other decision criteria
  while $com\_sit == ture$ **do**
    $t_{i\times j=1}$;
  end
  ...
end
// process all the obtained results, and then, derive a unique result based on priority
for $i = 0$ **to** $n - 1$ **do**
  for $j = 0$ **to** 8 **do**
    if $t_{i\times j} == 1$ **then**
      $a_{i\times j} = 1$;
      continue;
    end
  end
end

This algorithm's inputs are two two-dimensional matrices from the medical records and insurance information, and the output is a matrix with size $n \times 9$ since there are nine different decision outcomes. Firstly, the algorithm verifies if specific diseases exist in the input matrices. Then, it applies other decision rules to modify the corresponding values in the output matrix. It is important to note that not all rules are explicitly listed in the algorithm. Lastly, multiple modifications can occur in the result matrix during numerous judgments, so the algorithm identifies the first changed value in each row. Based on this information, the algorithm generates a new result matrix, yielding the correct decision outcome.

6. Analysis and Evaluation

In this section, we first provide a brief security analysis of our scheme. Then, we carry out a set of experiments to evaluate our decision-making scheme.

6.1. Security Analysis

Theorem 1. *Assuming that the Discrete Logarithm Problem (DLP) is hard in the Pedersen commitment group G and non-interactive zero-knowledge proofs, our proposed protocol is secure against up to* $n - 1$ *parties' collusion attack and guarantees the correctness validation.*

Proof. During the execution of the online phase protocol, we assumed there exists $\mathcal{A}$ controlling the participants. For the **Prepare**, **Input**, and **Compute** steps, since at most $n-1$ shares of the set for each value are uniformly random and do not reveal any information about the secret shares, it is impossible that $\mathcal{A}$ could tamper with the real transcript ϵ. During the output of *Result*, due to the information-theoretic hiding of our scheme, any of the values opened by the dishonest parties will be identified if the MAC checking fails. But, this happens with the possibility at most $2/p$ as proven in [9]. In the **Validate** step, assume that $\mathcal{A}$ changed the output y with another value that must open the commitment com_y correctly. But, in fact, we already obtained the correct commitment $\mathcal{W}(y, \bar{y})$ for y, and $\mathcal{A}$ can only calculate $\log_g h$. It is impossible to break based on the DLP.

Concerning the offline phase, the security of the **Prepare**, **GenMacKey**, **Input**, and **Triples** steps were proven exhaustively in [9]. Regarding the **Validate** step, it is also impossible for adversary $\mathcal{A}$ to break the commitments and NIZKPs since they are based on the difficulty of the DLP. Therefore, our protocol can be secure against collusion attacks by up to $n-1$ parties and guarantees correctness validation. □

6.2. Setup and Experimental Datasets

We ran our experiments on a laptop with an Intel(R) Core(TM) i5-12500H 2.50 GHz CPU, 16.0 GB RAM, and WSL 2 with the Ubuntu-22.04.2 LTS system in the LAN setting. The programming tools were Clion and Pycharm with C++ 17 and Python 3.10. We used a 128 bit prime as the computational modulus because this is a common choice, and the (statistical) security parameter was set to 40.

We collected two real-world datasets to conduct our experiments: patient medical records and insured person information. Tables 1 and 2 describe the variable names, types, and corresponding examples for these two datasets, respectively.

Table 1. The description of the medical center dataset.

Variable Name	Variable Type	Example
personnel ID	numerical	90209673
disease code	categorical	I10.X02
disease name	categorical	hypertension
total fees	numerical	7950.88
reimbursement amount	numerical	7189.04
settlement date	numerical	16 March 2022 10:06

Table 2. The description of the insurance company dataset.

Variable Name	Variable Type	Example
personnel ID	numerical	90209673
age	numerical	55
job hazard category	categorical	high
overweight condition	categorical	moderate
annual income	numerical	100,000
insurance premium	numerical	5000
insurance coverage	numerical	100,000
insurance type	categorical	hospitalization

6.3. Performance Evaluation

We compared the concrete performance of this paper's decision-making scheme with a normal one, i.e., choosing an arbitrary SMPC node as the validator without the HV (denoted as without-HV). To obtain the runtime of our solution, we implemented a benchmark that emulates both the online and offline phases. Notably, although the number of SMPC nodes had been set to 2 in the previous framework diagram, for this benchmark, we shall compare

the performance under different computational nodes, and their size in the range of 2–6 was set. For the experimental data to be statistically meaningful, we conducted the same experiment several times and took the mean values.

For the online phase, as illustrated in Figure 4, the runtime increased at a linear rate as the number of SMPC nodes grew, with the scheme without-HV or our scheme. When calculating in two nodes, it took approximately 19 ms to run the decision-making algorithm once using our proposed online protocol, and for without-HV, it took about 22 ms. It can be seen that, with an external validator to perform the verification work, the efficiency was roughly 1.14-times higher than the scheme without-HV. The follow-up multiparty scheme also remained at 1.1–1.2.

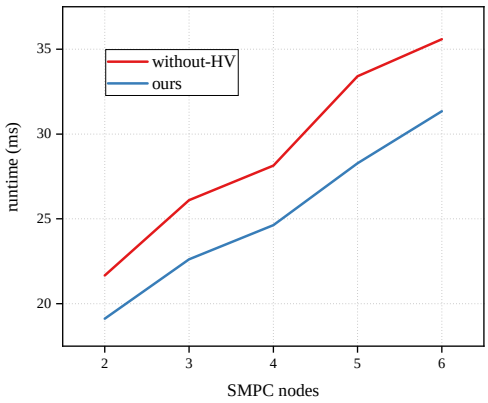

Figure 4. Time consumption for one-time online computation compared to the normal one without the HV (denoted as "without-HV"). The size of the computational nodes were set in the range of 2–6, and the runtime was measured in milliseconds.

Our primary concern in the preprocessing phase was the speed of triples' generation and ignoring the simultaneous generation of other examples such as squares (a, a^2), inverses (a, a^{-1}), and random bits. As shown in Figure 5, once the generated number was set to 4096, the execution time of the program tended to grow as the number of participants increased. For two parties, our proposed offline protocol took about 1.98 s. At the same time, the scheme without-HV was close to 2.49 s, almost 1.3-times more, attributed to the cumbersome verification and communication consumption.

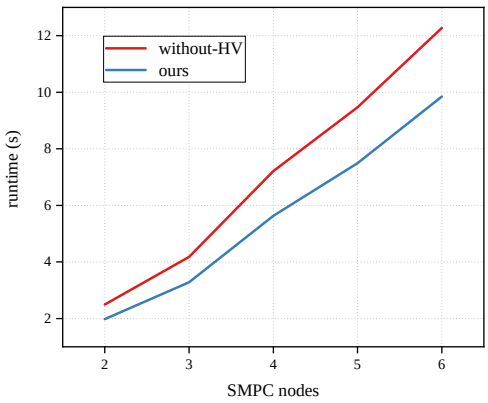

Figure 5. Time consumption for generating 4096 triples in the offline phase. Similarly, comparisons were made with the "without-HV" scheme, where the number of computational nodes ranged from 2 to 6, and the time was measured in seconds.

We also considered that, in real-world applications, if several users' decision-making requests need to be processed simultaneously, then there should be support for batch processing as well. Therefore, we prepared programs in an open-source platform to enable parallel computation. As depicted in Figure 6, we can see that the time consumption of the

calculation did not grow in a rather regular linear manner as the number of inputs increased due to the parallelization, which somewhat improved the efficiency. Our experiments found that the hardware used in this experiment can support simultaneous computation of up to approximately 20,000 inputs at a time.

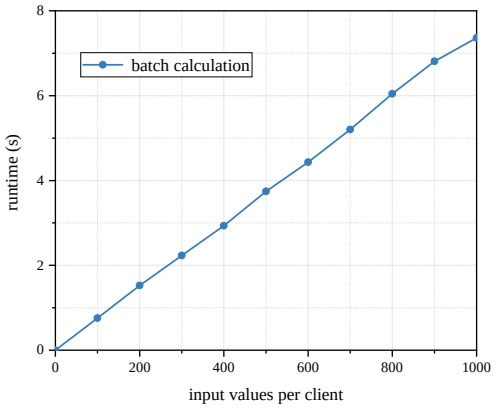

Figure 6. Execution time for parallel computation with different numbers of inputs in a two-SMPC-node setting.

Finally, as shown in Table 3, we gathered a set of meaningful metrics to display the overall performance of our scheme compared with without-HV in a two-node setting. We evaluated the disk consumption by the size of the compiled bytecode, transmission data, and preprocessing data volume. These metrics demonstrated that our scheme was acceptable for the disk consumption. In addition, the computation time of the two diverse schemes was 19.121 ms and 21.667 ms, respectively. This shows that our solution is efficient in terms of time consumption.

Table 3. Overall performance metrics of one-time online computation.

Type	Performance Metrics	Performance Data
our scheme	complied bytecode	369 KB
	communication round	80
	transmission data	0.050504 MB
	preprocessing data view	1562 triples + 3444 bit
	computational time	19.2127 ms
without-HV	complied bytecode	375 KB
	communication round	94
	transmission data	0.058192 MB
	preprocessing data view	2124 triples + 4688 bit
	computational time	21.667 ms

7. Conclusions and Future Work

In this paper, we first implemented an efficient decision-making scheme in the health insurance scenario using Secure Multiparty Computation. We introduced a novel framework diagram of our scheme by adding an external party (honest validator) to verify the correctness of the results. We extended the original SPDZ protocol and involved the Pedersen commitments to ensure correctness. Eventually, we implemented our scheme in the open-source framework MP-SPDZ, and in terms of the experimental results, our scheme was slightly more efficient than the ordinary one. In other words, it is feasible and practical. In addition, our scheme applies to general cross-institutional financial scenarios.

Although the offline phase can be accomplished before the evaluation, it is indisputable that the preprocessing of our solution was far worse than the original protocol. Especially in scenarios involving fewer participants, the triple generation speed was relatively slower than traditional approaches. Moreover, the time consumption of the online computation

also increased exponentially as the number of inputs grew in parallel. So, how to tackle the efficiency issue is what we need to solve desperately, and this is the direction of our future work.

Firstly, since the SPDZ protocol utilizes a linear secret-sharing scheme, we could contemplate employing a more-efficient one, such as the lattice-based commitment scheme BDLOP mentioned in [33]. It outperforms the Pedersen commitment scheme due to its linear homomorphic property, efficient ZKPs, and support for larger message space. Secondly, another more-straightforward approach involves leveraging high-performance GPUs [34] to enhance efficiency at the hardware level. Segregating computationally parallelizable portions into different grids for simultaneous computation—for instance, generating triples in the offline phase concurrently—might significantly augment the overall operational efficiency of the entire protocol.

Author Contributions: Conceptualization, T.W.; data curation, T.W.; formal analysis, T.W.; investigation, T.W.; methodology, T.W.; project administration, L.Z.; software, T.W.; supervision, Z.L. and Z.H.; visualization, T.W.; writing—original draft, T.W.; writing—review and editing, Z.L., Z.H. and L.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Key R&D Program of China (2021YFB3100700), the National Natural Science Foundation of China (62076125, 62032025, U20B2049, U20B2050, U21A20467, 62272228, U22B2029), the Shenzhen Science and Technology Program (JCYJ20210324134810028, JCYJ20210324134408023), the Key R&D Program of Guangdong Province (2020B0101090002), the Natural Science Foundation of Jiangsu Province (BK20200418), and the Shenzhen Virtual University Park Support Scheme (YFJGJS1.0).

Data Availability Statement: All data underlying the results are available as part of the article and no additional source data are required.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Pang, Z.H.; Fan, L.Z.; Guo, H.; Shi, Y.; Chai, R.; Sun, J.; Liu, G.P. Security of networked control systems subject to deception attacks: A survey. *Int. J. Syst. Sci.* **2022**, *53*, 3577–3598. [CrossRef]
2. Murthy, S.; Bakar, A.A.; Rahim, F.A.; Ramli, R. A comparative study of data anonymization techniques. In Proceedings of the 2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), Washington, DC, USA, 27–29 May 2019; IEEE: New York, NY, USA, 2019; pp. 306–309.
3. Aggarwal, G.; Feder, T.; Kenthapadi, K.; Motwani, R.; Panigrahy, R.; Thomas, D.; Zhu, A. Approximation algorithms for k-anonymity. *J. Priv. Technol.* **2005**, *2005112001*, 400.
4. Dwork, C. Differential privacy. In Proceedings of the Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, 10–14 July 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 1–12.
5. Li, C.; Yang, L.; Yu, S.; Qin, W.; Ma, J. SEMMI: Multiparty security decision-making scheme for linear functions in the internet of medical things. *Inf. Sci.* **2022**, *612*, 151–167. [CrossRef]
6. Damgård, I.; Damgård, K.; Nielsen, K.; Nordholt, P.S.; Toft, T. Confidential benchmarking based on multiparty computation. In Proceedings of the Financial Cryptography and Data Security: 20th International Conference, FC 2016, Christ Church, Barbados, 22–26 February 2016; Springer: Berlin/Heidelberg, Germany, 2017; pp. 169–187.
7. Kamm, L.; Willemson, J. Secure floating point arithmetic and private satellite collision analysis. *Int. J. Inf. Secur.* **2015**, *14*, 531–548. [CrossRef]
8. Catrina, O.; Saxena, A. Secure computation with fixed-point numbers. In Proceedings of the Financial Cryptography and Data Security: 14th International Conference, FC 2010, Tenerife, Canary Islands, 25–28 January 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 35–50.
9. Damgård, I.; Keller, M.; Larraia, E.; Pastro, V.; Scholl, P.; Smart, N.P. Practical covertly secure MPC for dishonest majority–or: Breaking the SPDZ limits. In Proceedings of the Computer Security–ESORICS 2013: 18th European Symposium on Research in Computer Security, Egham, UK, 9–13 September 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 1–18.
10. Damgård, I.; Pastro, V.; Smart, N.; Zakarias, S. Multiparty computation from somewhat homomorphic encryption. In Proceedings of the Advances in Cryptology–CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 643–662.
11. Keller, M. MP-SPDZ: A versatile framework for Multiparty computation. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Even, USA, 9–13 November 2020; pp. 1575–1590.

12. Yao, A.C.C. How to generate and exchange secrets. In Proceedings of the 27th Annual Symposium on Foundations of Computer Science (Sfcs 1986), Washington, DC, USA, 27–29 October 1986; IEEE: New York, NY, USA, 1986; pp. 162–167.
13. Goldreich, O. Secure multi-party computation. *Manuscript. Prelim. Vers.* **1998**, *78*.
14. Goldwasser, S. Multi party computations: Past and present. In Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing, Santa Barbara, CA, USA, 21–24 August 1997; pp. 1–6.
15. Cramer, R.; Damgård, I.; Escudero, D.; Scholl, P.; Xing, C. SPD$\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for Dishonest Majority. *IACR Cryptol. ePrint Arch.* **2018**, 482.
16. Keller, M.; Orsini, E.; Scholl, P. MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 830–842.
17. Keller, M.; Pastro, V.; Rotaru, D. Overdrive: Making SPDZ great again. In Proceedings of the Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, 29 April–3 May 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 158–189.
18. Dong, X.; Randolph, D.A.; Weng, C.; Kho, A.N.; Rogers, J.M.; Wang, X. Developing high performance Secure Multiparty Computation protocols in healthcare: A case study of patient risk stratification. *AMIA Summits Transl. Sci. Proc.* **2021**, *2021*, 200. [PubMed]
19. Garofalakis, M.N. Privacy Preserving Medical Data Analytics Using Secure Multi Party Computation. An End-to-End Use Case. Ph.D. Thesis, University of Athens, Athens, Greece, 2018.
20. Bogdanov, D.; Talviste, R.; Willemson, J. Deploying Secure Multiparty Computation for Financial Data Analysis: (Short Paper). In Proceedings of the Financial Cryptography and Data Security: 16th International Conference, FC 2012, Kralendijk, Bonaire, 27 Februray–2 March 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 57–64.
21. Cohen, J.D.; Fischer, M.J. A robust and verifiable cryptographically secure election scheme. In Proceedings of the 26th Annual Symposium on Foundations of Computer Science (sfcs 1985), Portland, OR, USA, 21–23 October 1985; IEEE Computer Society: Washington, DC, USA, 1985; pp. 372–382.
22. Adida, B. Helios: Web-based open-audit voting. In Proceedings of the 17th Conference on Security Symposium, San Jose, CA, USA, 28 July–1 August 2008; pp. 335–348.
23. Chaum, D.; Ryan, P.Y.; Schneider, S. A practical voter-verifiable election scheme. In Proceedings of the Computer Security–ESORICS 2005: 10th European Symposium on Research in Computer Security, Milan, Italy, 12–14 September 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 118–139.
24. Baum, C.; Damgård, I.; Orlandi, C. Publicly auditable Secure Multiparty Computation. In Proceedings of the Security and Cryptography for Networks: 9th International Conference, SCN 2014, Amalfi, Italy, 3–5 September 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 175–196.
25. Pedersen, T.P. Non-interactive and information-theoretic secure verifiable secret sharing. In Proceedings of the Advances in Cryptology—CRYPTO'91: Proceedings, Brighton, UK, 8–11 April 2001; Springer: Berlin/Heidelberg, Germany, 2001; pp. 129–140.
26. Cunningham, R.; Fuller, B.; Yakoubov, S. Catching MPC cheaters: Identification and openability. In Proceedings of the Information Theoretic Security: 10th International Conference, ICITS 2017, Hong Kong, China, 29 November–2 December 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 110–134.
27. Kanjalkar, S.; Zhang, Y.; Gandlur, S.; Miller, A. Publicly Auditable MPC-as-a-Service with succinct verification and universal setup. In Proceedings of the 2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Vienna, Austria, 6–10 September 2021; IEEE: New York, NY, USA, 2021; pp. 386–411.
28. Graf, M.; Küsters, R.; Rausch, D. AUC: Accountable Universal Composability. In Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 22–24 May 2023; IEEE: New York, NY, USA, 2023; pp. 1148–1167.
29. Bautista, O.G.; Akkaya, K.; Homsi, S. ReplayMPC: A Fast Failure Recovery Protocol for Secure Multiparty Computation Applications using Blockchain. In Proceedings of the 2023 IEEE International Conference on Smart Computing (SMARTCOMP), Nashville, TN, USA, 26–29 June 2023; IEEE: New York, NY, USA, 2023; pp. 124–132.
30. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009; pp. 169–178.
31. Beaver, D. Efficient multiparty protocols using circuit randomization. In Proceedings of the Advances in Cryptology—CRYPTO'91: Proceedings 11, Santa Barbara, CA, USA, 11–15 August 1992; Springer: Berlin/Heidelberg, Germany, 1992; pp. 420–432.
32. Camenisch, J.; Stadler, M. Efficient group signature schemes for large groups. In Proceedings of the Advances in Cryptology—CRYPTO'97: 17th Annual International Cryptology Conference Santa Barbara, CA, USA, 17–21 August 1997; Springer: Berlin/Heidelberg, Germany, 1997; pp. 410–424.
33. Rivinius, M.; Reisert, P.; Rausch, D.; Küsters, R. Publicly accountable robust Multiparty computation. In Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 23–25 May 2022; IEEE: New York, NY, USA, 2022; pp. 2430–2449.
34. Watson, J.L.; Wagh, S.; Popa, R.A. Piranha: A GPU platform for secure computation. In Proceedings of the 31st USENIX Security Symposium (USENIX Security 22), Boston, MA, USA, 10–12 August 2022; pp. 827–844.