Benchmarking Secure Multiparty Computation Frameworks for Real-World Workloads in Diverse Network Settings

Christopher Harth-Kitzerow[a], Jonas Schiller[b], Nina Schwanke[a], Thomas Prantl[b], Georg Carle[a]

[a] *Technische Universität München, Arcisstraße 21, München, 80333 , Bavaria, Germany*
[b] *Julius-Maximilians-Universität Würzburg, Am Hubland, Würzburg, 97074, Bavaria, Germany*

Abstract

Secure Multiparty Computation (MPC) enables distributed parties to jointly evaluate functions on their combined datasets while preserving individual data confidentiality. Although MPC protocols and frameworks have achieved significant performance improvements in recent years, particularly for complex workloads like secure neural network inference, systematic standardization and benchmarking of these frameworks remain underexplored.

This work comprehensively analyzes over 50 MPC applications to identify the core algorithmic structure most common in real-world MPC applications. From this analysis, we derive six reference use cases and implement these across four state-of-the-art MPC frameworks: HPMPC, MPyC, MP-SPDZ, and MOTION. We develop an open-source benchmarking framework[1] that evaluates these implementations under varying network conditions, including bandwidth constraints, latency, packet loss, and input sizes.

Our work presents the first systematic cross-framework evaluation of MPC performance based on real-world use cases across diverse network conditions and MPC security models. Thus, our comprehensive analysis yields novel insights into practical MPC performance and provides evidence-based recommendations for framework selection across different operational contexts.

Keywords: Secure Multiparty Computation, Use Case Study, Benchmark

1. Introduction

The volume of data produced by humanity is unprecedented and continues to grow at an exponential rate [1]. Although numerous open-source datasets exist, a significant amount of valuable data remains inaccessible due to privacy regulations, such as the General Data Protection Regulation in the EU or economic considerations [2]. A prime example where data transfer is prohibited by law is when two or more medical institutions want to analyze their joint patient data on a disease [3]. Yet, enabling this joint analysis could open up new insights, e.g., into the causes of certain diseases [4]. MPC is a cryptographic technique that enables multiple parties

[1] `https://github.com/chart21/mpcbench`

to jointly compute all computable functions on their private data while preserving the secrecy of the datasets [5]. Although the theoretical promises of MPC are immense, widespread adoption of the technology has not been achieved. While the reasons for this are manifold, we identify two main obstacles: (1) a lack of knowledge about potential use cases for MPC and (2) a lack of estimates regarding the performance overhead resulting from the use of MPC. Another complicating factor is that developers are challenged to choose from many possible MPC libraries, for which it is unclear how their performance relates to one another. Notably, MPC protocols require high-frequency network communication between distributed parties. Thus, network parameters such as latency, bandwidth, and packet loss have a major impact on MPC runtime. However, existing MPC performance analyses are limited and not standardized; for example, they differ in the workloads and network configurations considered. Therefore, the effect of different network configurations on MPC applications is underexplored.

We present MPCBench, an open-source MPC benchmarking framework designed to address these challenges for MPC libraries and workloads. MPCBench natively supports the four state-of-the-art general-purpose MPC frameworks MP-SPDZ [6], MOTION [7], HPMPC [8], and MPyC [9]. We further perform a comprehensive literature survey to implement six representative MPC workloads in MPCBench. To demonstrate the insights that developers can gain from MPCBench, we present extensive benchmark results that evaluate the MPC frameworks in various network scenarios.

2. Background

MPC protocols are designed to defend against different types of adversaries [10]. Semi-honest adversaries attempt to compromise the protocol's privacy without deviating from its prescribed steps, whereas malicious adversaries may seek to breach both privacy and correctness. Another critical aspect of an MPC protocol is its tolerance to adversarial corruption. Protocols that can withstand corruption of all but one participating party fall into the dishonest-majority category, while those that tolerate corruption of fewer than half the parties belong to the honest-majority category.

Cryptographic Building Blocks. Among the different cryptographic techniques to construct an MPC protocol, such as garbled circuits, function secret sharing, and linear secret sharing (LSS), LSS schemes have found the most widespread adoption in MPC frameworks [11, 12, 13, 14] due to their relatively low communication complexity. In LSS schemes, inputs are distributed to parties as so-called shares, which are uniformly random numbers in a field or ring of a fixed size. However, when holding a sufficient number of shares, a linear combination of these shares yields the original inputs. Some MPC protocols also provide a *Preprocessing Phase* that is input-independent and can be evaluated prior to the *Online Phase*, which handles input-dependent computation and communication.

Computation. Any computable function can be evaluated on secret shares by expressing the desired function as a circuit of boolean and arithmetic gates. The

circuit is then evaluated by the parties in a distributed manner. While jointly evaluating the circuit, some gates require network communication between parties before they can proceed to evaluate subsequent circuit layers. Hence, the runtime of MPC workloads benefits from low network latency when evaluating deep circuits and high network bandwidth when evaluating a large number of gates within a circuit layer. An often overlooked network property for MPC is packet loss, as resending a certain percentage of packets increases latency and decreases throughput when evaluating circuit layers.

3. Related Work

Although MPC protocol design improvements are regularly published [7, 15, 8], comparisons of protocol implementation have received little attention. There exists no standardized workload or benchmarking specification; thus, the considered applications, problem sizes, and network settings differ between publications. Only a few works include benchmarks for higher-level use cases: The authors of MP-SPDZ benchmark their implementation for one epoch of logistic regression and for the calculation of an inner product [6]. In MOTION, the authors benchmark computational primitives, such as the runtime of AND and OR gates, as well as potential use cases, including biometric matching [7]. The authors of the HPMPC framework evaluate its performance in various network network environments [8] and measure the number of evaluated gates per second and AES encryptions per second. They compare HPMPC's novel MPC protocols to other MPC protocols implemented in the same code base, as well as several other state-of-the-art frameworks. Furthermore, they evaluate the performance difference of the MPC protocols against a trusted third-party setting and the theoretically optimal runtime in a given network, when only considering the communication complexity. They show that HPMPC achieves up to 1 billion multiplications per second, while other frameworks perform at least two orders of magnitude fewer gates per second.

Maltitz et al. published the only work focused solely on benchmarking MPC by analyzing the performance of the general-purpose FRESCO framework [16]. The authors measure runtime, CPU cycles, memory consumption, and transmitted data volume, and vary the number of peers, transmission rate, packet loss, and latency for the use case of averaging distances traveled derived from cell phone GPS data. They find that increasing the number of participants linearly affects runtime and required RAM, while increasing network latency results in linear growth in runtime. Their results, however, are based on an arguably outdated framework and a single use case.

Hastings et al. [17] compare the current state of MPC compilers, which translate programs written by users in a high-level language into a representation that an MPC framework can evaluate. Their comparison is based solely on framework usability, and they refer to the original framework publications for all measurements. Ng and Chow analyze 53 recently published frameworks for privacy-preserving neural network training and inference [18]. They include 18 frameworks that rely on MPC. Their results show that if participants are willing to trust non-colluding servers, MPC solutions provide the fastest inference and training compared to other cryptographic

techniques, such as Homomorphic Encryption. However, the MPC frameworks suffer from an increase in inference time when the network bandwidth is reduced from 100 Mbit/s to 10 Mbit/s. Similarly, a higher latency in a WAN setting leads to slower inference and a lower total throughput of inferences per second. They also demonstrate that while some frameworks are optimized for low latency, meaning that the results for a single inference are computed efficiently, others are optimized for high throughput, meaning that the amortized inference time for a batch of inferences is minimized.

Use Cases and Collections. The first real-world implementation of MPC was a Danish sugar beet auction in 2008, where farmers participated in a auction to sell their products without revealing prices to any third party by using MPC [19]. In the following years, more real-world implementations followed, such as securely calculating the average wage gap between men and women in Boston and analyzing the dropout rates of computer science students in Estonia [20, 21]. In 2001, Du et al. presented a list of commonly appearing mathematical problems in MPC [22]. The list primarily consists of data analysis problems, including database queries and range searching. Talviste provides an overview of the most relevant MPC applications from 2015 [23]. However, this list is outdated, and no clustering or comparison between the use cases is performed.

In addition to these older lists, however, there are also more recent collections. For example, [24], which contains real-world data analysis use cases from national statistical offices up to 2023. Or [25], which also contains use cases up to 2023 while considering also other technologies besides MPC such as homomorphic encryption and differential privacy. The most current MPC application list is probably [25], which is maintained by the University of Berkeley at the time of writing this paper and contains 22 real-world use cases. In comparison, our use case list consists of over 60 use cases and is clustered more systematically than the existing ones.

4. Workloads

In this section, we present our extensive study of use cases, derived workloads, and experiment parameters.

4.1. Survey Methodology

As most real-world MPC deployments have emerged only recently, the number of public MPC use cases is small. Currently, there exist several use case collections, all far from complete [26, 24, 25] with researchers and NGOs curating these lists. We use these accumulations of use cases as the basis for our research. After removing duplicates, 30 different use cases remain in the collection. Additionally, we include more than 30 new use cases, resulting in a list of over 60 use cases, which is more than three times longer than previous collections [26].

We searched academic publications in two libraries, namely IEEE and ACM as two of the largest Computer Science libraries. For both libraries, we used the search terms "Secure Multiparty Computation" AND "Use Case" AND "Implementation" and "Secure Multiparty Computation" AND "Industry" AND "Implementation". This amounted to 84 + 98 results, although some of them were duplicates.

To identify industry-relevant use cases, we used a search engine for publications from (non)governmental organizations or companies, such as project reports or white papers. As it is unfeasible to work through all search engine results, we decided to work through the first 100 results for the search terms "Secure Multiparty Computation Use Case Implementation" and "Secure Multiparty Computation Industry", although there was considerable overlap between the two. The identified use cases were frequently found in blog posts published by the company that implemented them or through press releases aimed at promoting the adoption of new technologies.

This search yielded more than 200 results for academic papers and over 100 for industry reports, although some were duplicates. From these sources, we select only those that include a clear definition of the proposed benefit of using MPC, as well as a description of the implementation.

In the resulting collection of MPC use cases in science and industry, some use cases are similar. One example is the private inference using different neural networks. To keep our list concise, we have present one exemplary scenario in these cases, resulting in a total of 65 use cases.

4.2. Use Case Collection

Using the collected list, we cluster common MPC use cases and derive six representative workloads based on their underlying algorithmic structure. Table 1 shows an excerpt of the full use case collection with one example per cluster. The full use case collection is provided in the appendix in Table C.14. Each use case includes information on the security model, measured runtime in the original work, experiment parameters, and hardware used.

Table 1: Excerpt of studied use cases with one example per cluster

Cluster	Example					
	Description	Src.	Impl.	Adv.	Corrup. n	Runtime
Secure Statistics						
Home Loan Calculation: The government and public housing largely control Singapore's housing market. Certain factors must be met to rent an apartment from the government, such as a positive debt-to-income ratio and the ability to make a down payment. The proposed PoC allows the calculation of the loan terms without disclosing information to other parties.		[27]	PoC	Pass.	$> \frac{n}{2}$ 3	$\mathcal{O}(N)$
Secure Auction						
Sugar Beets: Establish a market-clearing price for sugar beets where farmers input their offers and the buyer inputs the quantity and purchase price. The input is then secretly shared across three servers performing the MPC.		[19]	Dep.	Pass.	$< \frac{n}{2}$ 3	1.2 h
Private Set Intersection						
Tax Fraud: Companies selling in Estonia are required to pay VAT to the state. The Ministry of Finance and Tax wants to compare sales and purchases to detect tax fraud. However, companies are concerned about revealing business-critical data by disclosing large sales and purchases. MPC enables tax fraud detection without disclosing confidential data.		[28]	PoC	Pass.	$< \frac{n}{2}$ 3	0.63 - 9 h
Logistic Regression						
Distributed Spam Filtering: A spam detection algorithm that trains on emails that are distributed between different parties to allow different companies to train their detectors together without disclosing sensitive information.		[29]	PoC	Pass.	$> \frac{n}{2}$ 2	4 s;1.14 h
Neural Network Inference						
Human Activity Recognition: Medical sensor data can quantify human behavior to analyze general health patterns. Private user data and the proprietary model are secured by MPC.		[30]	PoC	Pass.	both 2-3	n.e.
Threshold Cryptography						
Lightning Network IoT Bitcoin Payment: IoT devices participating in the Lightning Network (LN) secretly share private keys and commitments with the LN gateway to protect against collusion and ransom attacks.		[31]	PoC	Pass.	$> \frac{n}{2}$ 2	n.e.

PoC: Proof of Concept implementation, Dep: Real-World Deployment, n: Number of servers performing the MPC, $< \frac{n}{2}$: Honest Majority, $> \frac{n}{2}$: Dishonest Majority, n.e.: no explicit information

4.2.1. Number of Participants

Some MPC algorithms can perform a secure computation with an arbitrary number of participants. However, most research focuses on improving algorithms with two, three, or four participants. The use case list mirrors this, as most use cases are performed between two to four computation servers, with three being the most common setting. In some use cases, dozens [32, 20] or even thousands [19] of parties are required to submit secret inputs to the secure computation. The *outsourced computation model* [33] allows input parties that do not host an MPC server locally to create secret shares of their data and send one share to each computation server over an encrypted channel. This way, a set of three or four fixed servers can compute functions for an arbitrary number of input parties.

4.2.2. Network Setting

The quality of a network connection is often highly correlated with the distance between the servers [34]. The proof-of-concepts identified in the literature review have mostly been executed by geographically co-located companies and institutions within a single region or country. However, most use cases could also be extended to larger areas that span multiple countries. One example of this are medical use cases, such as a podiatrist collaboration to predict diseases [35], which could benefit from including more regions and, thus, more patients. As no use case implemented over a wide-area network provides information on measured latency, we turn to public sources on typical latency based on the distance between two servers. Based on a Verizon report, the typical round-trip time is about 13-14ms within Europe, 20-32 ms within the US, and 71 ms across the Atlantic [34]. Therefore, we vary the round-trip latency between 0.3 ms (co-located), 25 ms (same country), 50 ms (different countries), and 75 ms (different continents) in our benchmarks.

Few use cases report their connection bandwidth; however, a 100 Mbit/s connection is utilized in financial fraud detection using MPC [36], the sugar beets auction [19], and an intrusion detection use case [37]. While a 100 Mbit/s connection certainly is still common for home connections, modern servers frequently use connections with a bandwidth of several gigabits per second, especially when the servers are located within the same city. Therefore, we vary the bandwidth between 100 Mbit/s and 25 Gbit/s. The amount of packet loss is not mentioned for any use case in the list. Therefore, we assume general packet loss rates of 0%, 1%, 2%, and 5%. Although 5% is rare in server-to-server connections, it can occur in the case of mobile connections [38].

4.2.3. Computation Hardware

Many of the PoCs use local workstations to implement and benchmark their solutions [29, 37, 39]. In contrast, all real-world implementations, except the 2008 sugar beets auction, use server-grade hardware or cloud servers [21, 36]. The number of used cores ranges from 4 to 48, with RAM capacities ranging from 2 GB to 32 GB. Due to the continuous progress in hardware development over the last sixteen years, since the first real-world MPC use case, the range of hardware specifications in the use case list is rather large. Our experiments focus on modern server-grade hardware, yet we incorporate parameters to adjust the CPU quota and frequency in our benchmarking framework.

4.2.4. Performance Metrics

The most relevant metric for MPC applications is the execution time, also called runtime. MPC implementations are slower than plaintext computations, although the factor depends heavily on the chosen security setting, the specific use case, and the network connection between parties. Some MPC frameworks require laying out the entire computation as a circuit representation [40] or bytecode [6], thus requiring extensive memory. Another relevant metric is peak RAM utilization [16]. If the available system RAM is exceeded and swap memory is utilized, the runtime suffers noticeably [16], or the servers even crash. Another metric is the amount of data transmitted since it increases cost and runtime [41]. MPC protocols and applications should focus on minimizing the number of interaction rounds and the amount of total data sent between the parties. To obtain these metrics, we use the Linux *GNU time* command [42] to measure the execution time and peak RAM utilization of each framework. In addition to those framework-agnostic measurements, we also utilize the performance statistics reported by the different frameworks on metrics such as transmitted data.

4.3. Representative Use Case Clusters

Next, we describe the six most frequently encountered MPC use cases and the experiment parameters we derive for them to create workloads that mimic the demands of real-world MPC applications.

4.3.1. Secure Statistics

Secure statistics refers to use cases that computes the average, minimum, maximum, or other statistical values on secretly shared data. Two prominent use case examples of secure statistics from real life are the Boston wage gap study [20] and a study on economic development in Estonia [32]. In both studies, mean values were calculated across multiple data sets, including the average salaries of men and women. Examples of secure statistics are also found in academia, such as the privacy-preserving aggregation of network data for intrusion detection [37] and the calculation of statistics on household energy consumption [43]. Based on the existing secure statistics applications, we propose the following workloads: Two parties each have a data set of 10.000 32-bit integers and want to calculate the (i) mean value or (ii) minimum value of both data sets. However, computing an average is a trivial operation in MPC, with negligible overhead. Thus, we focus on case (ii) for our benchmark in section 6.

4.3.2. Secure Auctions

Auctioning enables sellers and buyers to submit confidential bids that specify the desired quantity and the price they are willing to pay for an item. Bids of sellers and buyers are then matched confidentially. Examples of MPC auctions in the real world include the Danish sugar beet auction [19], inventory trading by the American bank J.P. Morgan [44, 45] or CO_2-allowance trading in the aviation industry [46]. As a generalized workload for auctions, we chose the so-called double auction and assume, based on existing use cases, that there are a total of 500 bids and offers. Additionally, we assume the use of the client-server model, where the participating

parties secretly share their inputs with a small, fixed number of computation nodes before executing the MPC protocol. The prices are set discretely as positive integers between 1 and 100, and the quantity can vary between 1 and 10.000. The limited price range with a discrete set of values allows for more efficient implementations.

4.3.3. Private Set Intersect

Private Set Intersect (PSI) refers to obtaining the set of overlapping values in multiple datasets while preserving the confidentiality of all values not included in the output. In real life, MPC-based PSI is used, for example, on the website Callisto, which matches sexual predator incidents [47]. Other examples of MPC-based PSI include the "Students and Taxes" study, which matches the salaries and dropout rates of IT students in Estonia [48]. We propose the following use case as a generalized workload for MPC PSI: Two parties each have a dataset consisting of 100 IDs, where each ID is a 32-bit value. The PSI program involves determining the IDs that appear in both sets.

4.3.4. Logistic Regression Training

Real-world logistic regression applications have already been implemented using MPC. In the medical context, for example, MPC can perform training and inference of logistic regression models, which categorize patients at high risk for cardiovascular disease [49] or chronic heart disease [50], without requiring the sharing of any patient's data in plaintext. There are also MPC-based logistic regression use cases in academia, such as recognizing spam e-mails without sharing the e-mail in plain text [29]. Based on the existing use cases, we define our generalized workload for MPC logistic regression as follows: a batch-wise training of a logistic regression model that categorizes the entries of a dummy dataset with five features, 300 entries, and 100 training epochs. In this workload, two parties provide the horizontally split input to obtain the weights of the trained model. For dishonest-majoirty protocols we reduce the number of epochs to 10.

4.3.5. Neural Network Inference

In recent years, the state-of-the-art MPC frameworks MP-SPDZ, MPyC, and HPMPC have also enabled neural network inference [51, 52, 8], whereas MOTION does not provide a dedicated neural network interface. Additionally, two proof-of-concept examples have already been published that utilize MPC for private inference. In the first example, MRI scans are evaluated by neural networks [53], and the second one predicts drug-target interactions in medicine [54]. However, even more ideas are envisioned for MPC and neural networks, ranging from text classification and speech recognition to image classification [55]. Since, out of the considered frameworks, only HPMPC can perform inference on large neural networks (NN) and datasets, such as ResNet-50 on ImageNet, we focus on the batch-wise inference of smaller NN for our representative workload. Our workload performs inference of a (pre-trained) LeNet convolutional neural network model (CNN) on the MNIST dataset with a batch size of 64. Given the limitation to evaluating only small CNNs, we refer readers to the comprehensive benchmark study of private neural network computations [18] for larger models, which identified Piranha as the most efficient MPC-based framework using 3-4 computation nodes. Note that frameworks such

as Piranha can exclusively evaluate neural networks and are thus not considered general-purpose MPC frameworks, the focus of this work. Piranha [56] has since been superseded by the PIGEON framework [57], which integrates several system optimizations into HPMPC [8], including GPU support.

4.3.6. Threshold Cryptography

One of the most advanced industry use cases for MPC is called threshold cryptography. Multiple in-use systems exist for cryptocurrency wallets using MPC to protect their clients [58]. By using MPC, a private key can be partitioned into multiple shards and does not have to be stored in a single location. While users can sign their transfer by performing a secure distributed digital signature algorithm using the secretly shared keys, an adversary must gain access to multiple devices and servers to reconstruct the valid key. Additionally, MPC can also be used to protect symmetric keys or password hashes as implemented in the Sepior Key Management system [59]. Here, key servers are deployed in several cloud instances from different cloud providers, where each server stores a key share of a symmetric key. An adversary has to compromise multiple cloud providers to obtain the full key. Several financial institutions use the system to protect their data.

We use the AES algorithm for our generalized workload in the area of threshold security. For this workload, we evaluate 256 parallel encryptions of 128-bit blocks, as AES is often implemented and utilized in a batch-wise form to reduce encryption speed for larger plaintexts.

5. MPC Frameworks

Previous and deprecated MPC frameworks include Fairplay [60], Sharemind [19], Scale-Mamba [61], FRESCO [62], and ABY [63, 64, 15]. These frameworks have been surpassed by MP-SPDZ [51], MOTION [7], MPyC [9], and HPMPC [8] in terms of performance and usability. Table 2 provides an overview of these frameworks.

For completeness, we would like to mention that specialized MPC libraries are also available. One example is OpenMined PSI [65], which focuses on PSI calculation using MPC. Since these types of libraries are limited to very specific use cases, they can naturally make corresponding optimizations under the hood that are not scalable to implement in general-purpose MPC scenarios. We deliberately excluded these libraries from our benchmark because (1) they cannot be used to implement all of the scenarios collected, (2) we are convinced that libraries suitable for all scenarios are more important for the breakthrough of MPC in practice, and (3) there should be separate benchmarks for specialized tools.

5.1. MP-SPDZ

MP-SPDZ is a framework that combines more than 35 MPC protocols for various numbers of parties and adversary settings, with a focus on enabling fair comparisons across protocols and security settings [6]. It uses a Python programming interface for users that gets compiled into bytecode. The bytecode is then interpreted and executed by a C++-based virtual machine. This design choice enables developers to write code in a high-level language without explicitly triggering communication rounds between the parties. The framework supports all security models and allows

Table 2: MPC framework comparison

	Framework			
Detail	MP-SPDZ [51]	MOTION [7]	HPMPC [8]	MPyC [9]
Security Properties				
Passive Adversary	✓	✓	✓	✓
Active Adversary	✓	—	✓	—
Honest Majority	✓	—	✓	✓
Dishonest Majority	✓	✓	—	—
Implementation				
# MPC Protocols impl.	> 30 (Rep3, Yao, BMR, ...)	2 (GMW, BMR)	> 10 (ABY2, Trio, Quad, ...)	1 (Shamir's secret sharing)
Programming Language	C++, Python	C++	C++	Python
Separate Preprocessing	only for few protocols	—	✓	—
Documentation	✓	✓ (only in-code)	✓	✓
Strengths and Limitations				
Strengths	Supports all security models	Fast dishonest majority impl.	Highly efficient implementation	User-friendly interface for Data Science
Limitations	Overhead of virtual machine and bytecode	No fixed- or floating-point value support	No dishonest majority protocol support	Only one protocol supported

easy comparisons between algorithms across protocols within and across security groups. MP-SPDZ's compiler restructures the high-level instructions invoked by user-written programs to reduce the number of communication rounds required to evaluate a sequence of instructions. The bytecode supports operations on vectors of shares, and the virtual machines can evaluate several bytecodes with multiple CPU threads in parallel.

5.2. MOTION

MOTION is a general-purpose C++ framework for the semi-honest adversary setting with a dishonest majority, which aims to achieve low execution times by serializing communication [7]. MOTION supports two MPC protocols, Arithmetic and Boolean GMW [66], as well as Beaver-Micali-Rogaway BMR [67] and explicit switching between these protocols. It employs multiple techniques, including SIMD and multithreading, to enhance computational performance. Additionally, it is the only framework that explicitly serializes communication, eliminating the need for a dedicated TCP socket. Serialization transforms messages into packets ready for communication and adding appropriate metadata. This allows the use of low-overhead network protocols, such as UDP, which increases communication speed [7].

5.3. HPMPC

HPMPC implements more than 10 MPC protocols in C++ with a focus on computational efficiency [8]. HPMPC implements previously established protocols for performance comparisons, but focuses on the novel protocols proposed by the authors. The framework uses protocols tailored to specific settings, e.g., four-party protocols for malicious settings or three-party protocols for semi-honest settings. To parallelize local computation, the framework utilizes both SIMD instructions and Bitslicing. The framework does not rely on intermediate circuit representations or Bytecode like MP-SPDZ or MOTION, but instead overloads common C++ operators and functions with secure operations at compile time. The framework's lean design also compiles large MPC programs in a few seconds. To minimize the communication overhead, the framework interleaves communication and computation and MPC phases, such as the preprocessing and online phases.

5.4. MPyC

MPyC is a Python framework [9] that enables the easy implementation of MPC in a user-friendly way without an explicit focus on speed. It is based on arithmetic secret sharing and implements secure types and basic operations for integer, floating-point, and fixed-point numbers, enabling asynchronous evaluation through Python coroutines. MPyC is secure against a passive adversary in an honest majority setting. The framework also provides implementations for several algorithms, including neural network training and inference, linear programming, and decision tree training. The authors adapted the well-known Python library NumPy to increase computational speed when working with their secure types.

6. Benchmark

In this section, we present our benchmarking framework and the results of our workload evaluation. We perform all measurements on three servers equipped with an Intel(R) Xeon(R) Gold 6312U CPU @ 2.40GHz with 24 cores and 512 GB of RAM. Network configurations between parties are set using Linux's tc utility. For all frameworks, we aim to utilize the fastest possible settings, allowing for compilation improvements, multithreading, and share conversion. The appendix in section Appendix A.2 includes results for more than three servers and parties. Note that all figures presented in this section are shown in a more detailed fashion in section Appendix B.

6.1. Metrics

We begin by defining the metrics used to compare framework performance. Namely, the amount of data transferred D, peak RAM utilization R, and runtime T. We define the following variables: $P = \{P_i, ..., P_n\}$, $t_{i,j}$, r_i, $t_{i,s}$ and $t_{i,e}$. The set P comprises the involved parties $P_1 \dots P_n$. The total amount of data sent from P_i to P_j is referred to as $d_{i,j}$. We denote the maximum RAM utilization of P_i during protocol execution as r_i. For P_i, we denote with $t_{i,s}$ and $t_{i,e}$ the start of end time of protocol execution, respectively. Based on these variables we define $D = \sum_{i=1}^{|P|} \left(\sum_{j=1}^{|P|} d_{i,j} \right)$, $R = max\{r_1, \dots, r_{|P|}\}$ and $T = max\{(t_{1,e} - t_{1,s}), \dots, (t_{|P|,e} - t_{|P|,s})\}$. For all measurements, we report both mean and standard deviation after 10 repetitions.

6.2. MPCBench

Our open-source benchmarking framework, MPCBench[2], defines experiments through configuration files in pos-enabled testbeds [68], which consist of a management node and several worker nodes. A configuration file specifies the framework, workload, workload-based parameters such as input size and bit length, and network parameters such as bandwidth, packet loss, and network latency. Algorithm 1 illustrates how a configuration file is structured. The management node parses the configuration file and performs the following steps without requiring further user interaction.

1. The management node sets up several worker nodes (different servers) to install MPC frameworks and dependencies.

2. The worker nodes perform all possible experiment combinations for the specified workloads, network, and workload-related parameters. Network properties are set using Linux's *tc* utility.

3. For each combination, relevant metrics such as data communicated, runtime, and peak RAM usage are measured and exported to a CSV file.

4. Upon completion, the files are either uploaded to GitHub or stored on the management node.

Algorithm 1: Examplary experiment configuration

1: **Framework:** MOTION
2: **Experiment:** {Multiplication, Auction}
3: **Protocol:** Arithmetic GMW
4: **Input Size:** $\{100, 1000\}$
5: **Latency:** $\{0, 25, 50, 75\}$ ms
6: **Bandwidth:** $\{25000, 10000, 1000, 100\}$ Mbps
7: **Packet Drop:** $\{0, 1, 2, 5\}$ %

Out of the box, MPCBench supports the four discussed MPC frameworks and all six workloads. New workloads can be added by implementing them in the respective framework and including them in MPCBench's framework-specific setup scripts.

6.3. Evaluation of Primitive Operations

Before benchmarking the six workloads, we compare the frameworks' performance on the key primitive operation in MPC: Multiplications. Multiplication runtime indicates the general performance of an MPC framework. Our primitive benchmark securely multiplies 1.000.000 32-bit integers among three parties in a single communication round.

[2] `https://github.com/chart21/mpcbench`

6.3.1. Data transmitted

Tables 3 and 4 show multiple measurement results such as total communication, peak RAM consumption, and runtime in an optimal network setting (25 GBit/s, 0.3 ms latency, 0% packet loss). We observe that the data transmitted over the network differs by up to four times in the same security class across frameworks, but by more than one order of magnitude when transitioning from an honest-majority protocol to a dishonest-majority one. We also evaluate whether MPC frameworks can achieve the theoretical communication complexity of MPC protocols in practice. State-of-the-art 3PC protocols in the passive/honest-majority security class [69, 8] require three elements of total communication between the parties, as implemented by MP-SPDZ and HPMPC. Thus, the total data transmitted by the framework for the experiment should be 12 MB. Surprisingly, MP-SPDZ introduces an overhead of four times that of the theoretical optimum for field-based protocols.

6.3.2. Runtime

HPMPC outperforms all other frameworks for honest majority protocols and is more than one order of magnitude faster than the second-best framework. MPyC is the slowest passive/honest-majority framework, and is almost one order of magnitude slower than MP-SPDZ using the same protocol. For dishonest majority protocols, MP-SPDZ is approximately three times faster than MOTION in the same security model.

In MP-SPDZ, ring-based protocols tend to outperform field-based protocols by a slight margin, and dishonest protocols are approximately one order of magnitude slower than honest-majority protocols. On the other hand, adding active security to an honest majority protocol only introduces a slight overhead. All of these observations align with the theoretical communication complexity of the protocols.

Table 3: Comparison of protocols for 3-Party multiplication of 1,000,000 32-bit integers

Framework	Protocol	Adv.	Maj.	Com. (MB)	Runtime (s)	Network[a] Saturation (%)
HPMPC	Trio	passive	honest	**12.01**	$\mathbf{0.01 \pm 0.01}$	$\mathbf{27.5} \pm 9.8$[c]
MPyC	Shamir	passive	honest	20.00	3.14 ± 0.25	0.2 ± 0.02
MP-SPDZ	Rep.-Field	passive	honest	48.00	0.30 ± 0.02	5.0 ± 0.3
MP-SPDZ	Shamir	passive	honest	48.00	0.33 ± 0.02	4.6 ± 0.3
MP-SPDZ	Rep.-Ring	passive	honest	**12.00**	0.25 ± 0.02	1.5 ± 0.1
MP-SPDZ	PS-Rep.-Ring	active	honest	189.00	0.54 ± 0.04	11.2 ± 0.7
MP-SPDZ	Semi-2k	passive	dishonest	2370.22	$\mathbf{6.39} \pm 0.49$	$\mathbf{12.0 \pm 1.1}$
MOTION	GMW	passive	dishonest	**1865.49**	17.89 ± 1.76	3.3 ± 0.3

Bold font represents the best performance for each security model.

[a] Average network saturation over the entire execution to evaluate the computational performance of a framework.

[c] For larger input sizes, the framework exceeds 90% network saturation.

6.3.3. Ram Utilization

Table 4 shows the peak RAM utilization for the different frameworks. MOTION's RAM requirements exceed those of the other frameworks by more than one order of

Table 4: Peak RAM for 3-Party multiplication of 1,000,000 32-bit integers

Framework	Protocol	Adversary	Majority	Peak RAM (MB)	RAM/[b] Comm.
HPMPC	Trio	passive	honest	**12.66** $\pm$ 0.86	**1.05**
MPyC	Shamir	passive	honest	363.16 $\pm$ 8.14	18.16
MP-SPDZ	Rep.-Field	passive	honest	218.56 $\pm$ 2.40	4.55
MP-SPDZ	Shamir	passive	honest	178.22 $\pm$ 3.61	3.61
MP-SPDZ	Rep.-Ring	passive	honest	131.22 $\pm$ 1.81	10.94
MP-SPDZ	PS-Rep.-Ring	active	honest	591.55 $\pm$ 18.02	3.13
MP-SPDZ	Semi-2k	passive	dishonest	**477.00** $\pm$ 8.14	**0.20**
MOTION	GMW	passive	dishonest	15728.67 $\pm$ 1033.84	8.43

[b] Ratio of peak RAM consumption divided by total communication to evaluate resource efficiency of a framework.

magnitude. Thus, MOTION requires server-grade hardware with large amounts of RAM, even for simple programs. The smaller peak RAM utilization for MP-SPDZ's dishonest majority protocol is, therefore, due to the framework's implementation concept, not the security class choice. The RAM utilization for the active adversary protocol exceeds its semi-honest counterparts, likely due to the larger amount of cryptographic material required for execution. HPMPC requires more than one order of magnitude less RAM than other frameworks in the same security class.

The large-scale differences in performance between frameworks using protocols in the same security class may come as a surprise, given their similar theoretical communication complexity. The root cause is the design of the implementation. MP-SPDZ and MOTION use custom MPC compilers to interpret user-written functions. MOTION compiles the high-level program written by its users in C++ into an intermediate circuit representation, and MP-SPDZ compiles a high-level Python program into bytecode, which is then interpreted by a virtual machine written in C++. MPyC and HPMPC use compiler-less approaches. In MPyC, users employ high-level instructions on vectors in Python that are directly translated into their cryptographic instructions, as specified by the secret-sharing protocol. Similarly, HPMPC allows users to write high-level programs based on custom data types in C++, which overload operators and routines with any implemented MPC protocol using template-based inheritance. From a performance perspective, the compiler-less approach is superior in terms of runtime and memory overhead. Note that by using Python rather than a low-level programming language, MPyC also introduces additional memory overhead compared to the other frameworks.

6.3.4. Network Saturation

From the data on communication, runtime, and peak memory, we derive two useful measurements: network saturation and the ratio of peak RAM to the protocol's communication complexity. Since a given protocol sending k Mbit of data over a link with a bandwidth of b Mbit/s can theoretically at best achieve a runtime of $\frac{k}{b}$ seconds, network saturation puts a strict lower bound on the runtime that an implementation of that protocol can achieve in practice. Especially workloads requiring only a small number of communication rounds, such as the presented multiplica-

tion workload, should achieve high network saturation. However, most frameworks achieve less than 10% network saturation on the given workload. This suggests a large room for implementation-related improvements to accelerate local computation. The only frameworks achieving notable network saturation are HPMPC and, in the dishonest-majority class, MP-SPDZ. The difference in network saturation between honest-majority and dishonest-majority MP-SPDZ is likely explained by the efficient oblivious transfer (OT) implementation that MP-SPDZ utilizes, which is only required for preprocessing in dishonest-majority protocols. As HPMPC is optimized for highly parallelizable workloads, increasing the workload's input size to more than one million results in over 90% network saturation, leaving little room for further implementation-related improvements.

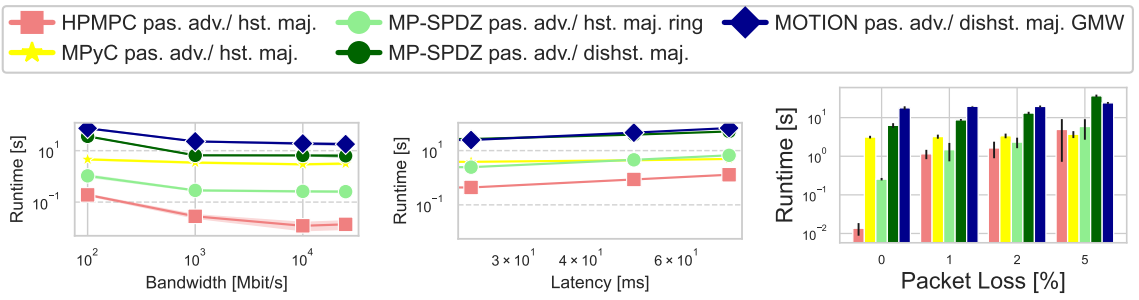
6.3.5. Resource Efficiency

Calculating the ratio of peak RAM required by a framework to the protocol's communication complexity is useful for evaluating a framework's memory requirements. A high ratio indicates that the framework adds potentially unnecessary overhead to the expected RAM requirements. In contrast, a low ratio around 1 suggests that a framework manages memory efficiently by even cleaning up the memory of completed communication rounds. Again, out of the different frameworks and protocols, only HPMPC and dishonest-majority MP-SPDZ achieve a good result, with ratios close to 1. Again, the difference between the different MP-SPDZ protocols is likely attributed to the efficient OT implementation, as OTs account for the majority of communication complexity in dishonest majority MPC. HPMPC achieves low memory requirements by sending and receiving data in fixed-size buffers and allocating memory only for a minimal number of buffers at any given time. MOTION's large overhead can be explained by its circuit representation, which models each gate. In contrast, MPyC's memory overhead is likely due to using a Python backend instead of a C++ backend.

6.3.6. Variation of Network Parameters

While evaluations in optimal network settings demonstrate the peak performance of MPC frameworks, MPC workloads are often deployed in settings with geographically distributed parties, ranging from those in the same city to those on different continents. A range of network settings should be considered to systematically evaluate the performance of different frameworks. Fig. 1 shows the runtime for the multiplication benchmark with varying latencies, bandwidths, and packet loss on a logarithmic scale. Overall, the different network parameters have a similar impact on the frameworks' performance. When manipulating the bandwidth between the parties, the runtime drops most notably from 100Mbit/s to 1000Mbit/s, while only HPMPC shows a noticeable improvement when more than 1Gbit/s is available, suggesting that the other frameworks are constrained by local processing power. Increasing the network latency between parties to up to 75ms impacts performance linearly for each framework. Increasing packet loss to just 1% equalizes performance across the framework, as all honest-majority protocols and all dishonest-majority protocols show highly similar performance, suggesting that the network is the main bottleneck rather than the local computation conducted by each framework. Overall, MP-SPDZ performs better than MOTION for the dishonest majority except for

network settings with with high latency and packetloss, while HPMPC consistently performs best for the honest majority protocols.



(a) Bandwidth (b) Latency (c) Packetloss

Figure 1: Multiplication benchmark in various network settings

Protocols: Trio (HPMPC), Shamir (MPyC), Rep3/Semi2k (MP-SPDZ), GMW (MOTION)

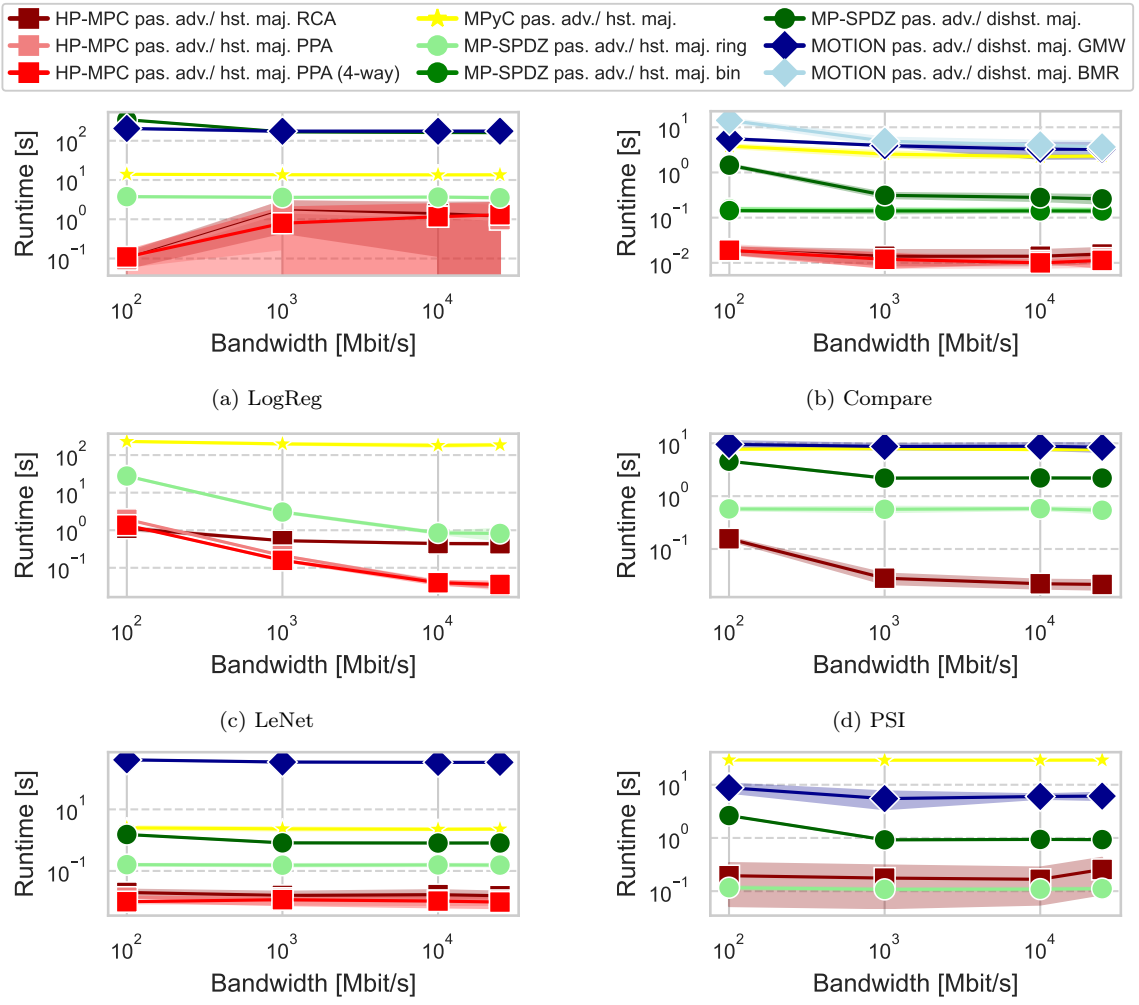
6.4. Bandwidth Evaluations

Figure 3 illustrates the performance of the frameworks for the six different workloads with varying network bandwidths between the parties. For experiments involving share conversion between the arithmetic and boolean domain, or experiments that are evaluated exclusively in the boolean domain, we add framework-specific tweaks and protocols such as utilizing different kinds of adder circuits during share conversion (Ripple Carry Adder (RCA), Parallel Prefix Adder (PPA), PPA with multi-input AND gates (PPA 4-Way), or using protocols that work well for boolean circuits specifically (binary secret sharing (bin), BMR). Note that we only include these additional variations if the workload benefits from them. Also, some experiments do not include all frameworks or protocols. For instance, neural network inference is too slow or even crashes for dishonest majority protocols and is thus evaluated only for honest majority protocols.

We observe that the relative ranking of the frameworks remains mainly consistent. For the honest majority class, HPMPC performs best, while MpyC performs worse than MP-SPDZ. For the dishonest majority class, MP-SPDZ tends to perform better than MOTION. MOTION even exhausts the 512GB of available memory for neural network inference (LeNet).

The impact of increasing bandwidth above 100 Mbps on framework performance is best noticeable for large workloads and fast frameworks, suggesting that most frameworks do not achieve sufficient performance of local operations to utilize additional network bandwidth. This can be especially observed in the PSI workload, where only HPMPC's runtime improves with increased available network bandwidth. Notably, when choosing the right framework, each workload can be solved within a few seconds, even in settings with low network bandwidth. Only training the logistic regression model in the dishonest majority setting takes more time, while neural network inference with dishonest-majority protocols is not feasible within a realistic time frame.

Overall, the results suggest that, in contrast to theoretical expectations, most frameworks are unable to utilize fast network bandwidths due to computational



(a) LogReg

(b) Compare

(c) LeNet

(d) PSI

(e) Auction

(f) AES

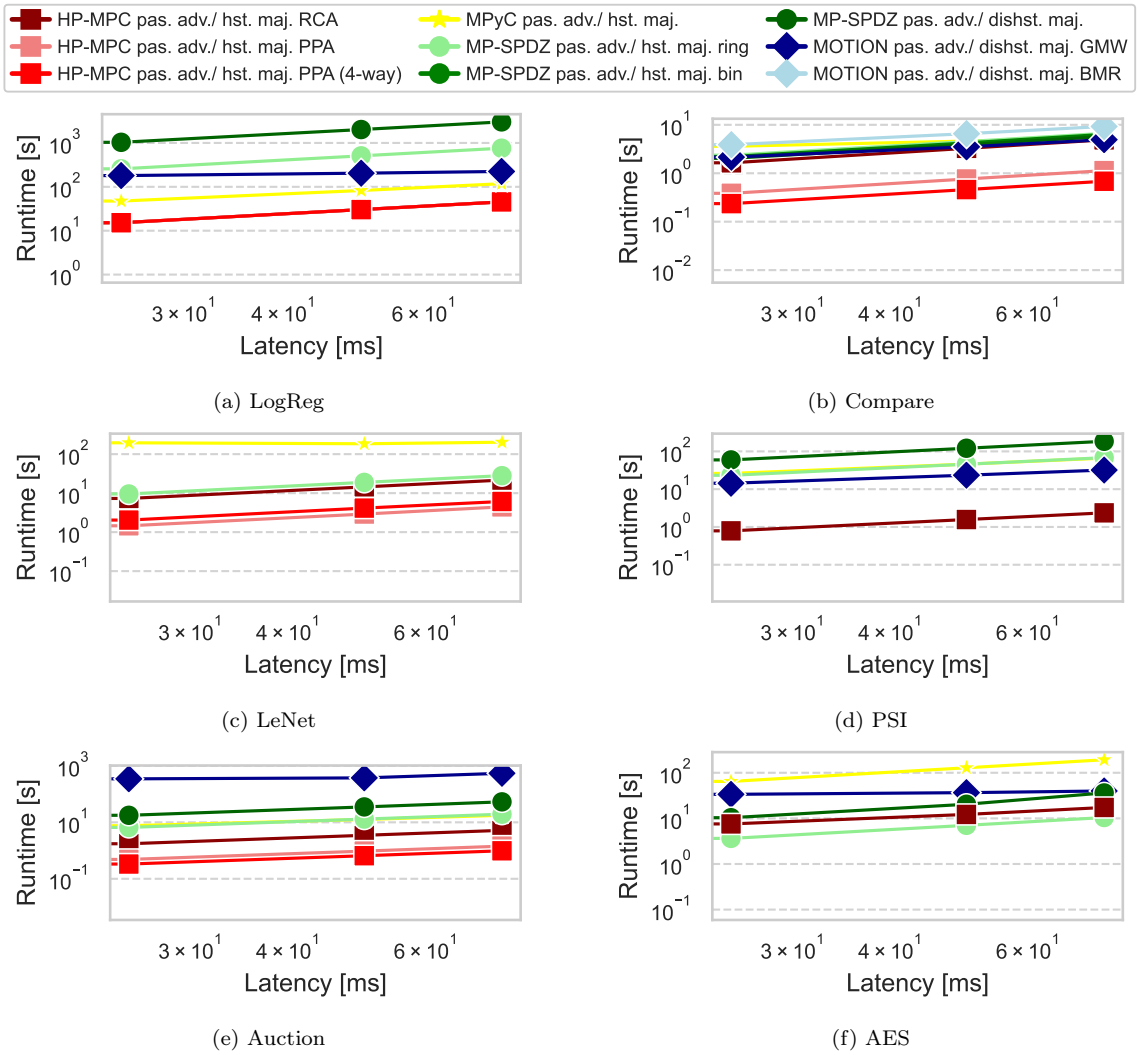
Figure 3: Bandwidth comparison across different MPC frameworks and protocols

Protocols: Trio (HPMPC), Shamir (MPyC), Rep3/Semi2k (MP-SPDZ), GMW/BMR (MOTION)

overhead or a lack of acceleration. Outliers with increased runtime under higher bandwidth have high standard deviation and are deemed measurement errors.

6.5. Latency Evaluations

Figure 5 shows the performance of the frameworks for the six workloads with different network latencies between the parties. Again, the ranking of the best framework remains consistent, with the only exception being dishonest majority protocols in high latency settings where MOTION performs better than MP-SPDZ for 4/5 workloads. In line with theoretical communication complexity, increasing network latency linearly increases the protocol runtime for all frameworks and protocols. Even with high latencies of 75ms between the parties, there is a significant difference between framework runtimes in the same security class. As the number of communication rounds is similar between protocols for the different workloads, the results suggest that even in latency-constrained environments, a framework's implementation-related efficiency is important.



(a) LogReg

(b) Compare

(c) LeNet

(d) PSI

(e) Auction

(f) AES

Figure 5: Latency comparison across different MPC frameworks and protocols

Protocols: Trio (HPMPC), Shamir (MPyC), Rep3/Semi2k (MP-SPDZ), GMW/BMR (MOTION)

6.6. Packetloss Evaluations

Figure 7 illustrates the performance of the frameworks for various workloads with different percentages of packet loss between parties. While for most workloads the relative ranking is again consistent, increasing packet loss tends to shrink the runtime gap between frameworks in the same security class as the runtime becomes constrained by the network rather than the frameworks' implementation efficiency. The faster frameworks HPMPC and MP-SPDZ in the honest majority setting are most affected by packet loss, as their implementation is fast enough to saturate a large proportion of the network. Slower frameworks remain largely unaffected. As the difference between no packet loss and small packet loss has a significant impact on the runtime of the faster framework, there might be room for implementation-related improvements. For instance, replicating messages on lossy networks in advance might avoid the latency costs of retransmitting messages.

6.7. RAM Utilization

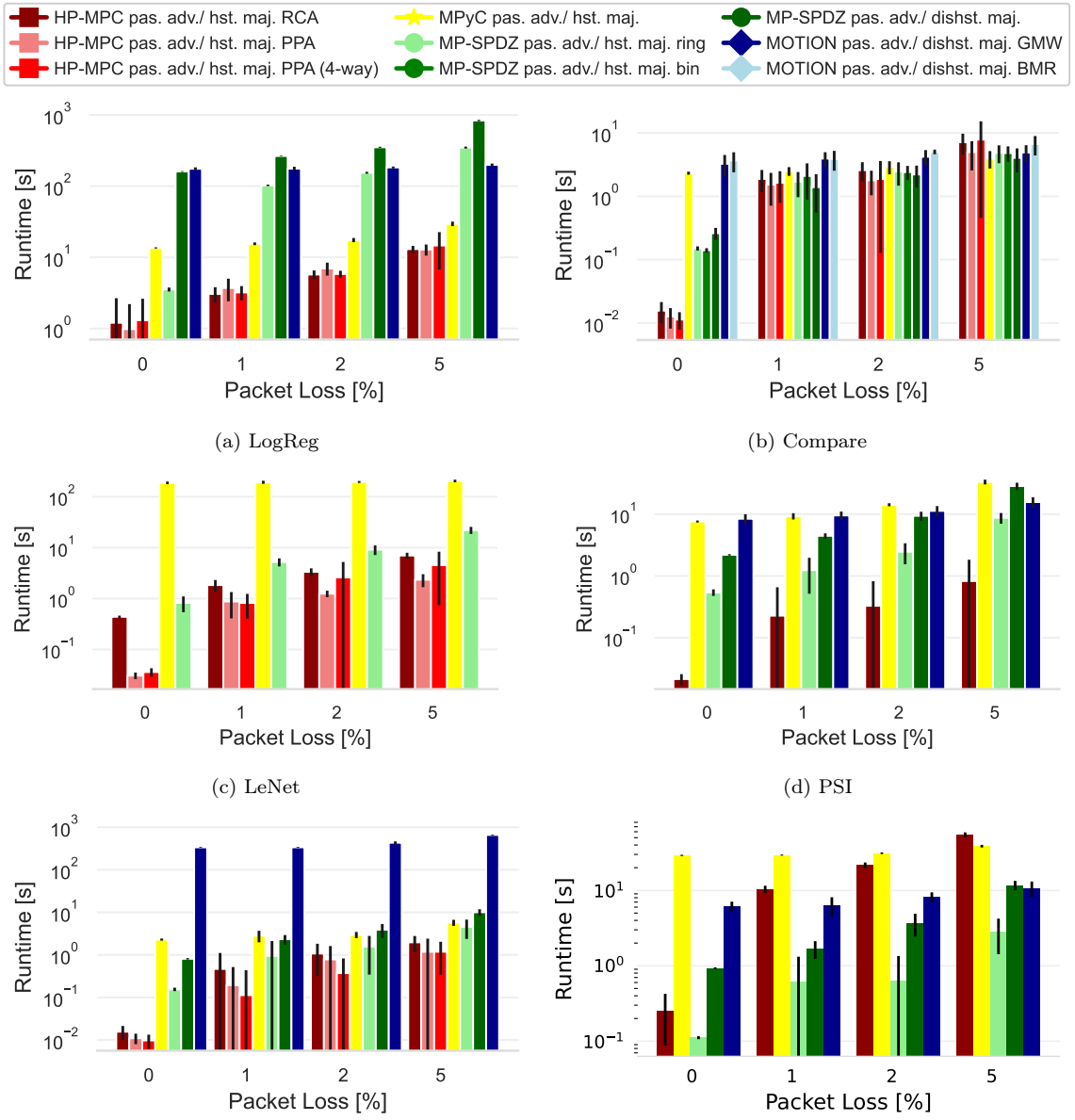
Table 5 and 6 show the peak RAM utilization of an individual party for the different frameworks and different workloads. Overall, the RAM utilization differs by up to three orders of magnitude within the same security class. MOTION especially requires large amounts of memory, exceeding 70GB when evaluating the Auction use case. Among the honest majority frameworks, HPMPC requires only a few MB of RAM for each workload, demonstrating the advantages of a compiler-less framework and a lightweight C++ implementation.

Table 5: Peak memory usage comparison [MB] across Honest-Majority protocols

Use Case	HPMPC (Trio)	MPyC (Shamir)	MP-SPDZ (Rep3)
Compare	3.13 ± 0.58	158.07 ± 8.05	51.46 ± 1.30
Auction	3.90 ± 0.47	117.29 ± 8.18	18.30 ± 0.90
PSI	6.30 ± 0.60	199.16 ± 8.12	49.55 ± 0.83
LogReg	4.01 ± 0.20	343.31 ± 12.55	75.3 ± 2.72
LeNet	15.76 ± 1.62	4703.12 ± 55.76	527.3 ± 2.83

Table 6: Peak memory usage comparison [MB] across Dishonest-Majority protocols

Use Case	MP-SPDZ (Semi-2K)	MOTION (GMW)	MOTION (BMR)
Compare	252.1 ± 9.64	448.76 ± 7.50	1071.10 ± 25.20
Auction	79.67 ± 3.16	78624.40 ± 15038.00	—
PSI	105.22 ± 5.47	1218.80 ± 6.50	—
LogReg	1082.20 ± 14.03	41291.96 ± 253.74	—
LeNet	—	—	—



(a) LogReg

(b) Compare

(c) LeNet

(d) PSI

(e) Auction

(f) AES

Figure 7: Packetloss comparison across different MPC frameworks and protocols

Protocols: Trio (HPMPC), Shamir (MPyC), Rep3/Semi2k (MP-SPDZ), GMW/BMR (MOTION)

6.8. Decision Tree

Our results are summarized in a decision tree shown in Figure 9, which supports developers unfamiliar with MPC in choosing the best general-purpose framework for their requirements. It is based on the key differentiation criteria of MPC protocols, as well as the assumed security setting, e.g., honest vs. dishonest majority and malicious vs. semi-honest adversary. The other decisions are based on the need for fast execution and customization. Although MPyC provides well-structured documentation and functionality familiar to Python programmers, its slower speed prohibits its use in scenarios with high-performance requirements. For the dishonest majority protocols, only MOTION and MP-SPDZ provide an implementation. MP-SPDZ outperforms MOTION in the majority of tested use cases and settings. Also MOTION's lack of fixed-point value computation, high RAM requirement, and inadequate documentation make it more challenging to use than MP-SPDZ. Only in network settings with high latency or packetloss MOTION can be considered a viable alternative in case the previously mentioned hurdles are not a concern. HPMPC is the fastest available general-purpose framework for the honest-majority setting. It's also the only framework that is able to handle large workloads, i.e. neural network inference with large batch size or multiple GB of shared AES encryptions without crashing. This is achieved by its compiler-less approach and an interleaved preprocessing model that synchronizes the preprocessing phase and the online phase to only store a small proportion of preprocessing material in RAM at any point.

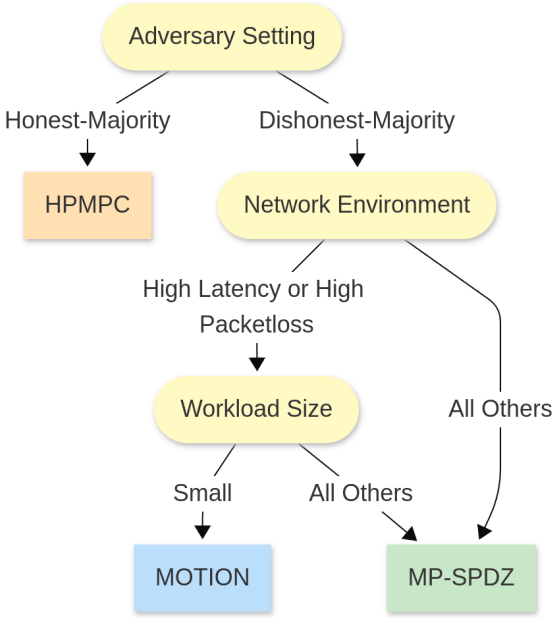


Figure 9: Decision tree for selecting a general-purpose MPC framework

7. Conclusion

In this work, we present an open-source benchmark for general-purpose Secure Multiparty Computation (MPC) frameworks. To define a set of representative workloads, we study more than 60 MPC deployments and identify six representative MPC

use cases. We find that real-world PoCs and implementations mostly assume semi-honest adversaries. Also, there are more PoCs relying on an honest majority than a dishonest majority security assumption, while three servers are typically used to carry out the computation. Most PoCs assume a client-server setting, allowing for an arbitrary number of parties to submit input to a fixed number of computation servers. Given these typical settings, we performed extensive evaluations of workloads in four general-purpose MPC frameworks: MP-SPDZ, MOTION, HPMPC, and MpyC.

While runtimes differ greatly between frameworks, the relative ranking for each workload and network environment remains consistent. HPMPC outperforms other frameworks in the honest majority setting, often by more than one order of magnitude. MP-SPDZ tends to outpferfom MOTION in the dishonest majority setting with the exception of network settings with high latency and packetloss. MPyC could not compete with the speed of the other frameworks. The relative difference between frameworks remains high even in slow networks. We find that these differences emerge due to implementation efficiency rather than different communication complexity between implementations. Currently only HPMPC is able to run large workloads without crashing by using a compiler-less approach to evaluate circuits and efficient memory management across MPC phases.

Future work may also analyze the effect of larger numbers of participating servers on the different use cases. This could be enhanced by varying network parameters only for specific links. In real-world networks, link speeds vary greatly based on the location of the servers. This could be emulated by selecting different network settings for each connecting link between the parties. We also raise an open question: Are there use cases where MPC compilers reduce runtime not only in theory but also in practice when considering the performance overhead observed in our benchmark caused by evaluating compiled program representations?

Overall, our findings suggest that based on their current performance, state-of-the-art MPC frameworks are ready for widespread adoption and can solve real-world use cases in practical time frames. The large difference in runtime between frameworks, sometimes for the same MPC protocols, motivates further optimizations from an implementation- rather than a protocol perspective.

Acknowledgements

Large Language Models were utilized to edit text for this work. After using these services, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article. This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] Statista, Worldwide: volume of data/information created, captured, copied and consumed from 2010 to 2023, with forecasts to 2028 (in zettabytes), `https://www.statista.com/statistics/871513/worldwide-data-created/`.

[2] U. N. C. of Experts on Big Data, D. S. for Official Statistics, Privacy-enhancing technologies for official statistics (2023).

[3] X. Dong, D. A. Randolph, C. Weng, A. N. Kho, J. M. Rogers, X. Wang, Developing high performance secure multi-party computation protocols in healthcare: A case study of patient risk stratification, AMIA Summits Transl. Sci. Proc. 2021 (2021) 200–209.

[4] D. Evans, V. Kolesnikov, M. Rosulek, A pragmatic introduction to secure multi-party computation, Foundations and Trends® in Privacy and Security 2 (2–3) (2018) 70–246. doi:10.1561/3300000019.

[5] Y. Lindell, Secure multiparty computation, Communications of the ACM 64 (1) (2020) 86–96. doi:10.1145/3387108.

[6] M. Keller, Mp-spdz: A versatile framework for multi-party computation, Proceedings of the ACM Conference on Computer and Communications Security (2020) 1575–1590doi:10.1145/3372297.3417872.

[7] L. Braun, D. Demmler, T. Schneider, O. Tkachenko, Motion-a framework for mixed-protocol multi-party computation, ACM Transactions on Privacy and Security 25 (2022). doi:10.1145/3490390.

[8] C. Harth-Kitzerow, A. Suresh, Y. Wang, H. Yalame, G. Carle, M. Annavaram, High-throughput secure multiparty computation with an honest majority in various network settings, 25th Privacy Enhancing Technologies Symposium (PETS 2025) (2025).

[9] B. Schoenmakers, Mpyc—python package for secure multiparty computation, in: Workshop on the Theory and Practice of MPC. https://github.com/lschoe/mpyc, 2018.

[10] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, Y.-a. Tan, Secure multiparty computation: Theory, practice and applications, Information Sciences 476 (2019) 357–372. doi:10.1016/j.ins.2018.10.024.

[11] T. Araki, J. Furukawa, Y. Lindell, A. Nof, K. Ohara, High-Throughput Semi-Honest Secure Three-Party Computation with an Honest Majority, in: CCS, 2016.

[12] H. Chaudhari, A. Choudhury, A. Patra, A. Suresh, ASTRA: High Throughput 3PC over Rings with Application to Secure Prediction, in: CCSW@CCS, 2019.

[13] A. P. K. Dalskov, D. Escudero, M. Keller, Fantastic Four: Honest-Majority Four-Party Secure Computation With Malicious Security, in: USENIX Security, 2021.

[14] H. Chaudhari, R. Rachuri, A. Suresh, Trident: Efficient 4PC Framework for Privacy Preserving Machine Learning, in: NDSS, 2020.

[15] A. Patra, T. Schneider, A. Suresh, H. Yalame, Aby2.0: Improved mixed-protocol secure two-party computation (full version) *, 2020.

[16] M. von Maltitz, G. Carle, A performance and resource consumption assessment of secure multiparty computation, in: International Workshop on Cryptocurrencies and Blockchain Technology, 2018. doi:10.1007/978-3-030-00305-0_25.
URL http://arxiv.org/abs/1804.03548http://dx.doi.org/10.1007/978-3-030-00305-0_25

[17] M. Hastings, B. Hemenway, D. Noble, S. Zdancewic, Sok: General purpose compilers for secure multi-party computation, IEEE, 2019, pp. 1220–1237. doi:10.1109/SP.2019.00028.

[18] L. K. L. Ng, S. S. M. Chow, Sok: Cryptographic neural-network computationf, in: 2023 IEEE Symposium on Security and Privacy (SP), 2023, pp. 497–514. doi:10.1109/SP46215.2023.10179483.

[19] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. Schwartzbach, T. Toft, Secure multiparty computation goes live (2008).
URL http://sikkerhed.alexandra.dk/uk/projects/simap

[20] A. Lapets, N. Volgushev, A. Bestavros, F. Jansen, M. Varia, Secure mpc for analytics as a web application, in: 2016 IEEE Cybersecurity Development (SecDev), 2016, pp. 73–74. doi:10.1109/SecDev.2016.027.

[21] D. Bogdanov, L. Kamm, B. Kubo, R. Rebane, V. Sokk, R. Talviste, Students and taxes: a privacy-preserving study using secure computation, Proceedings on Privacy Enhancing Technologies 2016 (2016) 117–135. doi:10.1515/popets-2016-0019.

[22] W. Du, M. J. Atallah, Secure multi-party computation problems and their applications: A review and open problems (2001).

[23] R. Talviste, Practical Applications of Secure Multiparty Computation, Vol. 13, IOS Press, 2015, pp. 246–252.

[24] D. Buckley, 16. united nations pet lab: International trade (2 2023).
URL https://unstats.un.org/wiki/display/UGTTOPPT/16.+United+Nations+PET+Lab%3A+International+Trade

[25] Repository of use cases - pet adoption guide (2023).

[26] D. Kaviani, Mpc deployments (2023).

[27] F. M. Chan, Q. Xu, H. J. Seah, S. L. Keoh, Z. Tang, K. M. M. Aung, Privacy preserving computation in home loans using the fresco framework, in: K. Arai, S. Kapoor, R. Bhatia (Eds.), Advances in Information and Communication Networks, Springer International Publishing, Cham, 2019, pp. 90–107.

[28] Marko, S. Sander, V. M. B. Dan, J. oemets, How the estonian tax and customs board evaluated a tax fraud detection system based on secure multi-party computation, Springer Berlin Heidelberg, 2015, pp. 227–234.

[29] M. A. Pathak, M. Sharifi, B. Raj, Privacy preserving spam filtering (2011). arXiv:1102.4021.

[30] D. Melanson, R. Maia, H. Kim, A. C. A. Nascimento, M. D. Cock, Secure multi-party computation for personalized human activity recognition, Neural Process. Lett. 55 (3) (2023) 2127–2153. doi:10.1007/S11063-023-11182-8.

[31] A. Kurt, S. Mercan, O. Shlomovits, E. Erdin, K. Akkaya, Lngate: powering iot with next generation lightning micro-payments using threshold cryptography, in: Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 117–128. doi:10.1145/3448300.3467833.

[32] Riivo, W. J. B. Dan, Talviste, Deploying secure multi-party computation for financial data analysis, Springer Berlin Heidelberg, 2012, pp. 57–64.

[33] I. Damgård, K. Damgård, K. Nielsen, P. S. Nordholt, T. Toft, Confidential benchmarking based on multiparty computation, in: J. Grossklags, B. Preneel (Eds.), Financial Cryptography and Data Security, Springer Berlin Heidelberg, Berlin, Heidelberg, 2017, pp. 169–187.

[34] VERIZON, Ip latency statistics (2024).
URL https://www.verizon.com/business/terms/latency/

[35] R. Labs, Breaktrough in preventive care (2023).
URL https://www.podotherapie.nl/blog/item/2023/07/13/Doorbraak-in-preventieve-voetzorg

[36] A. Sangers, M. van Heesch, T. Attema, T. Veugen, M. Wiggerman, J. Veldsink, O. Bloemen, D. Worm, Secure multiparty pagerank algorithm for collaborative fraud detection, Springer-Verlag, 2019, pp. 605–623. doi:10.1007/978-3-030-32101-7_35.

[37] M. Burkhart, M. Strasser, D. Many, X. Dimitropoulos, Sepia: Privacy-preserving aggregation of multi-domain network events and statistics, USENIX Association, 2010, p. 15.

[38] J. Lloret, L. Parra, M. Taha, J. Tomás, An architecture and protocol for smart continuous ehealth monitoring using 5g, Computer Networks 129 (05 2017). doi:10.1016/j.comnet.2017.05.018.

[39] B. Hemenway, S. Lu, R. Ostrovsky, W. W. IV, High-precision secure computation of satellite collision probabilities, 2016, pp. 169–187. doi:10.1007/978-3-319-44618-9_9.

[40] L. Braun, D. Demmler, T. Schneider, O. Tkachenko, MOTION - A Framework for Mixed-Protocol Multi-Party Computation, ACM Trans. Priv. Secur. (2022).

[41] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, K. Seth, Practical secure aggregation for privacy-preserving machine learning, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 1175–1191. doi:10.1145/3133956.3133982.

[42] A. Gordon, Gnu time (6 2018).
URL https://www.gnu.org/software/time/

[43] C. Thoma, T. Cui, F. Franchetti, Secure multiparty computation based privacy preserving smart metering system, in: 2012 North American Power Symposium (NAPS), 2012, pp. 1–6. doi:10.1109/NAPS.2012.6336415.

[44] A. Polychroniadou, G. Asharov, B. Diamond, T. Balch, H. Buehler, R. Hua, S. Gu, G. Gimler, M. Veloso, Prime match: A privacy-preserving inventory matching system, https://eprint.iacr.org/2023/400 (2023).
URL https://eprint.iacr.org/2023/400

[45] S. Mazloom, B. E. Diamond, A. Polychroniadou, T. Balch, An efficient data-independent priority queue and its application to dark pools, Proc. Priv. Enhancing Technol. 2023 (2) (2023) 5–22. doi:10.56553/POPETS-2023-0038.
URL https://doi.org/10.56553/popets-2023-0038

[46] M. Zaninyz, D. Perez, T. T. Delibasiz, G. Inalhan, M. Fidanoglu, E. A. Pereira, V. Mirchandani, E. Koyuncu, I. Ozkol, A. Enrich, J. C. Triana, C. Pasaolu, Enabling the aviation co2 allowance trading through secure market mechanisms, SIDs 2014 - Proceedings of the SESAR Innovation Days (11 2014).

[47] A. Rajan, L. Qin, D. W. Archer, D. Boneh, T. Lepoint, M. Varia, Callisto: A cryptographic approach to detecting serial perpetrators of sexual misconduct, in: Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies, COMPASS '18, Association for Computing Machinery, New York, NY, USA, 2018. doi:10.1145/3209811.3212699.

[48] D. Bogdanov, M. Jõemets, S. Siim, M. Vaht, How the Estonian Tax and Customs Board Evaluated a Tax Fraud Detection System Based on Secure Multi-party Computation, Springer Berlin Heidelberg, 2015, p. 227–234. doi:10.1007/978-3-662-47854-7_14.

[49] M. Veeningen, S. Chatterjea, A. Horváth, G. Spindler, E. Boersma, P. Van der Spek, O. van der Galien, J. Gutteling, W. Kraaij, T. Veugen, Enabling analytics on sensitive medical data with secure multi-party computation, Studies in health technology and informatics 247 (2018) 76–80.

[50] Inpher, Augmented heart disease analysis (2020).
URL https://inpher.io/solutions/by-industry/healthcare/

[51] M. Keller, E. Orsini, P. Scholl, Mascot: Faster malicious arithmetic secure computation with oblivious transfer, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 830–842. doi:10.1145/2976749.2978357.

[52] B. Schoenmaker, Mpyc package (2023).

[53] A. Santamaria-Pang, Leverage secure multi party computation (smpc) for machine learning inference in rs-fmri datasets. (2 2024).
URL https://techcommunity.microsoft.com/t5/healthcare-and-life-sciences/leverage-secure-multi-party-computation-smpc-for-machine/ba-p/4057703

[54] B. Hie, H. Cho, B. Berger, Realizing private and practical pharmacological collaboration, Science 362 (6412) (2018) 347–350. arXiv:https://www.science.org/doi/pdf/10.1126/science.aat4807, doi:10.1126/science.aat4807.
URL https://www.science.org/doi/abs/10.1126/science.aat4807

[55] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, L. van der Maaten, Crypten: Secure multi-party computation meets machine learning (2022). arXiv:2109.00984.

[56] J.-L. Watson, S. Wagh, R. A. Popa, Piranha: A {GPU} platform for secure computation, in: 31st USENIX Security Symposium (USENIX Security 22), 2022, pp. 827–844.

[57] C. Harth-Kitzerow, Y. Wang, R. Rajat, G. Carle, M. Annavaram, Pigeon: A framework for private inference of neural networks, 25th Privacy Enhancing Technologies Symposium (PETS 2025) (2025).

[58] O. Shlomovits, Introducing multi-party ecdsa library (11 2019).

[59] 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 61 (2018) 1749–1771. doi:10.1093/comjnl/bxy090.

[60] D. Malkhi, N. Nisan, B. Pinkas, Y. Sella, Fairplay—A secure Two-Party computation system, in: 13th USENIX Security Symposium (USENIX Security 04), USENIX Association, San Diego, CA, 2004.
URL https://www.usenix.org/conference/13th-usenix-security-symposium/fairplay{\T1\textemdash}-secure-two-party-computation-system

[61] A. Aly, B. Coenen, K. Cong, K. Koch, M. Keller, D. Rotaru, O. Scherer, P. Scholl, N. P. Smart, T. Tanguy, T. Wood, Scale-mamba (10 2023).
URL https://homes.esat.kuleuven.be/~nsmart/SCALE/

[62] Alexandra Institute, FRESCO - a FRamework for Efficient Secure COmputation, https://github.com/aicis/fresco.

[63] D. Demmler, T. Schneider, M. Zohner, Aby - a framework for efficient mixed-protocol secure two-party computation, Internet Society, 2015. doi:10.14722/ndss.2015.23113.

[64] P. Mohassel, P. Rindal, Aby3: A mixed protocol framework for machine learning, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 35–52. doi:10.1145/3243734.3243760.

[65] OpenMined, PSI: Private Set Intersection (ECDH + Bloom / Golomb), https://github.com/OpenMined/PSI, accessed: 18. November 2025 (2025).

[66] O. Goldreich, S. Micali, A. Wigderson, How to play any mental Game-A completeness theorem for protocols with honest majority, in: Proceedings of the 19th Annual ACM Symposium on Theory of Computing, New York, 1987, pp. 218–229.

[67] D. Beaver, S. Micali, P. Rogaway, The round complexity of secure protocols, in: Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing, STOC '90, Association for Computing Machinery, New York, NY, USA, 1990, p. 503–513. doi:10.1145/100216.100287.

[68] S. Gallenmüller, D. Scholz, H. Stubbe, G. Carle, The pos framework: a methodology and toolchain for reproducible network experiments, in: Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies, 2021, pp. 259–266.

[69] T. Araki, J. Furukawa, Y. Lindell, A. Nof, K. Ohara, High-throughput semi-honest secure three-party computation with an honest majority, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 805–817. doi:10.1145/2976749.2978331.

[70] D. Beaver, Efficient multiparty protocols using circuit randomization, in: J. Feigenbaum (Ed.), Advances in Cryptology — CRYPTO '91, Springer Berlin Heidelberg, Berlin, Heidelberg, 1992, pp. 420–432.

[71] M. O. Rabin, How to exchange secrets with oblivious transfer, IACR Cryptol. ePrint Arch. 2005 (2005) 187.
URL https://api.semanticscholar.org/CorpusID:15222660

[72] S. Becker, P. Duplys, J. Graf, K. Graffi, A. Grassi, D. Greven, J. Grewe, S. Jain, T. Klenk, N. Matyunin, H. Modica, V. Raskin, P. Scherer, V. Suschke, S. Trieflinger, V. Vlasakiev, J. Weinfurtner, Carbyne stack. doi:10.5281/zenodo.8192460.
URL https://carbynestack.io

[73] Findings use case monitoring human trafficking (4 2022).
URL https://datasharingcoalition.eu/app/uploads/2022/04/220420-final-deliverable-use-case-monitoring-human-trafficking.pdf

[74] M. Zanin, E. A. Pereira, V. Mirchandani, A. Enrich, J. C. Triana, Design and implementation of a secure auction system for air transport slots, in: 2015 IEEE World Congress on Services, 2015, pp. 160–166. doi:10.1109/SERVICES.2015.32.

[75] Governance design for mpc data collaboration (2023).
URL https://coe-dsc.nl/wp-content/uploads/2023/06/governance-for-an-mpc-data-collaboration-based-on-elderly-care-monitoring-case.pdf

[76] Proving the value potential of mpc for maas (2022).
URL https://coe-dsc.nl/use-cases/advancing-data-collaboration-for-monitoring-the-dutch-elderly-care-through-mpc-technology/

[77] A. Lapets, F. Jansen, K. D. Albab, R. Issa, L. Qin, M. Varia, A. Bestavros, Accessible privacy-preserving web-based data analysis for assessing and addressing economic inequalities, in: Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies, COMPASS '18, Association for Computing Machinery, New York, NY, USA, 2018. doi:10.1145/3209811.3212701.

[78] H. Cho, D. J. Wu, B. Berger, Secure genome-wide association analysis using multiparty computation, Nature Biotechnology 36 (2018) 547–551. doi:10.1038/nbt.4108.

[79] M. Ion, B. Kreuter, E. Nergiz, S. Patel, M. Raykova, S. Saxena, K. Seth, D. Shanahan, M. Yung, Private intersection-sum protocols with applications to attributing aggregate ad conversions, in: 2020 IEEE European Symposium on Security and Privacy (EuroS&P), 2020, pp. 370–389.
URL https://eprint.iacr.org/2019/723.pdf

[80] Inpher, Human resources attrition analysis.
URL https://inpher.io/solutions/by-use-case/hr-attrition-analysis/

[81] Inpher, Fraud data collaboration (2020).
URL https://inpher.io/solutions/by-industry/financial-services/

[82] J. Feigenbaum, B. Pinkas, R. Ryger, F. Saint-Jean, Secure computation of surveys, 2004, pp. 2–14.

[83] Inpher, Clinical trials patient selection and matching (2020).
URL https://inpher.io/solutions/by-industry/healthcare/

[84] M. Liu, L. Cheng, Y. Gu, Y. Wang, Q. Liu, N. E. O'Connor, Mpc-csas: Multi-party computation for real-time privacy-preserving speed advisory systems, IEEE Transactions on Intelligent Transportation Systems 23 (2022) 5887–5893. doi:10.1109/TITS.2021.3052840.

[85] S. Straus, A federal government privacy-preserving technology demonstration, Georgetown University (6 2021).

[86] A. Mitriaieva, E. Weitenberg, Statistics netherlands: Measuring effectiveness of an ehealth solution using private set intersection (11 2021).

[87] B. Scheenstra, A. Bruninx, F. van Daalen, N. Stahl, E. Latuapon, M. Imkamp, L. Ippel, S. Duijsings-Mahangi, D. Smits, D. Townend, I. Bermejo, A. Dekker, L. Hochstenbach, M. Spreeuwenberg, A. M. Jos, van 't Hof, B. Kietselaer, Digital health solutions to reduce the burden of atherosclerotic cardiovascular disease proposed by the carrier consortium, JMIR Cardio 6 (2022) e37437.

[88] Steven, N. A. G. Rosario, Goldfeder, Threshold-optimal dsa/ecdsa signatures and an application to bitcoin wallet security, Springer International Publishing, 2016, pp. 156–174.

[89] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, T. Toft, Privacy-preserving face recognition, in: I. Goldberg, M. J. Atallah (Eds.), Privacy Enhancing Technologies, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 235–253.

[90] D. Gupta, A. Segal, A. Panda, G. Segev, M. Schapira, J. Feigenbaum, J. Rexford, S. Shenker, A new approach to interdomain routing based on secure multi-party computation, Association for Computing Machinery, 2012, pp. 37–42. doi:10.1145/2390231.2390238.

[91] F. Zobiri, M. Gama, S. Nikova, G. Deconinck, A privacy-preserving three-step demand response market using multi-party computation, in: 2022 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), 2022, pp. 1–5. doi:10.1109/ISGT50606.2022.9817546.

[92] D. Roman, K. Vu, Enabling data markets using smart contracts and multi-party computation, in: W. Abramowicz, A. Paschke (Eds.), Business Information Systems Workshops, Springer International Publishing, Cham, 2019, pp. 258–263.

[93] Inpher, Predictive maintenance for distributed fleets.
URL https://inpher.io/solutions/by-use-case/defense-iot/#predictive-maintenance

[94] M. Vono, Pets in advertising: Scenarios for secure multi-party computation (5 2023).
URL https://medium.com/criteo-engineering/pets-in-advertising-scenarios-for-secure-multi-party-computation-4d2e667e5ecd

[95] S. Atapoor, N. P. Smart, Y. T. Alaoui, Private liquidity matching using mpc, in: S. D. Galbraith (Ed.), Topics in Cryptology – CT-RSA 2022, Springer International Publishing, Cham, 2022, pp. 96–119.

[96] E. Marquet, J. Moeyersons, E. Pohle, M. V. Kenhove, A. Abidin, B. Volckaert, Secure key management for multi-party computation in mozaik, in: 2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), 2023, pp. 133–140. doi:10.1109/EuroSPW59978.2023.00020.

[97] O. L. Team, Oasis labs partners with meta to assess fairness for its ai models using cutting-edge privacy technologies (7 2022).
URL https://medium.com/oasislabs/oasis-labs-partners-with-meta-to-assess-fairness-for-its-ai-models-97e153d707ef

[98] M. Abspoel, Improving hiv treatment choice with multi-party computation (9 2021).
URL https://www.cwi.nl/documents/195230/slides%20Mark%20Abspoel.pdf

[99] A. Resende, D. Railsback, R. Dowsley, A. C. A. Nascimento, D. F. Aranha, Fast privacy-preserving text classification based on secure multiparty computation, IEEE Transactions on Information Forensics and Security 17 (2022) 428–442. doi:10.1109/TIFS.2022.3144007.

[100] L. Li, A. Bayuelo, L. Bobadilla, T. Alam, D. A. Shell, Coordinated multi-robot planning while preserving individual privacy, in: 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 2188–2194. doi:10.1109/ICRA.2019.8794460.

[101] M. Breuer, P. Hein, L. Pompe, B. Temme, U. Meyer, S. Wetzel, Solving the kidney exchange problem using privacy-preserving integer programming, in: 2022 19th Annual International Conference on Privacy, Security & Trust (PST), 2022, pp. 1–10. doi:10.1109/PST55820.2022.9851968.

[102] M. Barni, P. Failla, R. Lazzeretti, A.-R. Sadeghi, T. Schneider, Privacy-preserving ecg classification with branching programs and neural networks, IEEE Transactions on Information Forensics and Security 6 (2) (2011) 452–468. doi:10.1109/TIFS.2011.2108650.

[103] G. Asharov, T. Balch, A. Polychroniadou, Privacy-preserving portfolio pricing, in: Proceedings of the Second ACM International Conference on AI in Finance, ICAIF '21, Association for Computing Machinery, New York, NY, USA, 2022. doi:10.1145/3490354.3494400.

Appendix A. Additional Evaluations

In this section, we present additional benchmarks when measuring the preprocessing and online phases separately, as well as when increasing the number of computation servers from three to four parties.

Appendix A.1. Effects of Separating Preprocessing/Online Phase

Table A.7: Preprocessing, online, and external runtime in seconds for Private Set Intersection with 100 values for each party and different latencies (Passive Adversary)

Framework	Runtime Type	Latency (ms)			
		0	25	50	75
HPMPC hst. maj.	Preprocess (s)	0.01 ± 0.00	0.26 ± 0.04	0.46 ± 0.07	0.66 ± 0.12
	Online (s)	0.01 ± 0.00	0.64 ± 0.05	1.23 ± 0.13	1.93 ± 0.23
	External (s)	0.03 ± 0.01	1.11 ± 0.06	2.18 ± 0.13	3.26 ± 0.21
MP-SPDZ hst. maj.	Preprocess (s)	0.01 ± 0.00	0.23 ± 0.00	0.43 ± 0.00	0.62 ± 0.05
	Online (s)	0.28 ± 0.05	17.85 ± 0.02	35.27 ± 0.03	52.72 ± 0.04
	External (s)	0.47 ± 0.05	20.16 ± 0.02	39.58 ± 0.04	59.02 ± 0.02
MP-SPDZ dishst. maj.	Preprocess (s)	35.98 ± 0.13	450.96 ± 0.14	865.80 ± 0.06	1281.85 ± 0.25
	Online (s)	0.40 ± 0.01	35.39 ± 0.09	72.01 ± 0.14	107.80 ± 0.32
	External (s)	36.63 ± 0.13	488.13 ± 0.08	941.11 ± 0.15	1394.46 ± 0.46
MOTION dishst. maj.	Preprocess (s)	0.37 ± 0.01	1.26 ± 0.02	2.11 ± 0.02	3.00 ± 0.04
	Online (s)	4.90 ± 1.14	12.29 ± 0.92	20.33 ± 0.81	27.91 ± 0.75
	External (s)	8.43 ± 1.57	14.37 ± 0.93	23.44 ± 0.82	32.08 ± 0.74

Protocols: Trio (HPMPC), Rep3/Semi2k (MP-SPDZ), BMR (MOTION)

Table A.8: Preprocessing, online, and external runtime in seconds for Private Set Intersection with 100 values for each party and different latencies (Passive Adversary)

Framework	Runtime Type	Latency (ms)			
		0	25	50	75
HPMPC hst. maj.	Preprocess (s)	0.01 ± 0.00	0.26 ± 0.04	0.46 ± 0.07	0.66 ± 0.12
	Online (s)	0.01 ± 0.00	0.64 ± 0.05	1.23 ± 0.13	1.93 ± 0.23
	External (s)	0.03 ± 0.01	1.11 ± 0.06	2.18 ± 0.13	3.26 ± 0.21
MP-SPDZ hst. maj.	Preprocess (s)	0.01 ± 0.00	0.23 ± 0.00	0.43 ± 0.00	0.62 ± 0.05
	Online (s)	0.28 ± 0.05	17.85 ± 0.02	35.27 ± 0.03	52.72 ± 0.04
	External (s)	0.47 ± 0.05	20.16 ± 0.02	39.58 ± 0.04	59.02 ± 0.02
MP-SPDZ dishst. maj.	Preprocess (s)	35.98 ± 0.13	450.96 ± 0.14	865.80 ± 0.06	1281.85 ± 0.25
	Online (s)	0.40 ± 0.01	35.39 ± 0.09	72.01 ± 0.14	107.80 ± 0.32
	External (s)	36.63 ± 0.13	488.13 ± 0.08	941.11 ± 0.15	1394.46 ± 0.46
MOTION dishst. maj.	Preprocess (s)	0.37 ± 0.01	1.26 ± 0.02	2.11 ± 0.02	3.00 ± 0.04
	Online (s)	4.90 ± 1.14	12.29 ± 0.92	20.33 ± 0.81	27.91 ± 0.75
	External (s)	8.43 ± 1.57	14.37 ± 0.93	23.44 ± 0.82	32.08 ± 0.74

Protocols: Trio (HPMPC), Rep3/Semi2k (MP-SPDZ), BMR (MOTION)

Table A.9: Preprocessing, online, and external runtime in seconds for Private Set Intersection with 100 values for each party and different bandwidths

Framework	Runtime Type	Bandwidth (Mbit/s)			
		100	1000	10000	25000
HPMPC hst. maj.	Preprocess (s)	0.06 ± 0.02	0.01 ± 0.00	0.01 ± 0.00	0.01 ± 0.00
	Online (s)	0.13 ± 0.01	0.02 ± 0.00	0.01 ± 0.00	0.01 ± 0.00
	External (s)	0.25 ± 0.01	0.05 ± 0.01	0.03 ± 0.01	0.03 ± 0.01
MP-SPDZ hst. maj.	Preprocess (s)	0.03 ± 0.00	0.02 ± 0.00	0.02 ± 0.00	0.01 ± 0.00
	Online (s)	0.37 ± 0.04	0.30 ± 0.07	0.31 ± 0.04	0.28 ± 0.05
	External (s)	0.57 ± 0.05	0.49 ± 0.07	0.50 ± 0.04	0.47 ± 0.05
MP-SPDZ dishst. maj.	Preprocess (s)	46.69 ± 0.57	37.08 ± 0.12	35.94 ± 0.16	35.98 ± 0.13
	Online (s)	0.51 ± 0.03	0.42 ± 0.01	0.40 ± 0.01	0.40 ± 0.01
	External (s)	47.46 ± 0.56	37.76 ± 0.13	36.60 ± 0.16	36.63 ± 0.13
MOTION dishst. maj.	Preprocess (s)	1.77 ± 0.10	0.49 ± 0.01	0.37 ± 0.01	0.37 ± 0.01
	Online (s)	4.72 ± 0.62	4.86 ± 0.84	5.05 ± 0.76	4.90 ± 1.14
	External (s)	9.54 ± 1.46	8.71 ± 1.03	8.79 ± 1.29	8.43 ± 1.57

Protocols: Trio (HPMPC), Rep3/Semi2k (MP-SPDZ), BMR (MOTION)

Table A.10: Preprocessing, online, and external runtime in seconds for Private Set Intersection with 100 values for each party and different packet loss

Framework	Runtime Type	Packet Loss (%)			
		0	1	2	5
HPMPC hst. maj.	Preprocess (s)	0.01 ± 0.00	0.01 ± 0.00	0.21 ± 0.42	0.53 ± 0.81
	Online (s)	0.01 ± 0.00	0.12 ± 0.33	0.22 ± 0.43	0.31 ± 0.39
	External (s)	0.03 ± 0.01	0.44 ± 0.72	1.05 ± 0.83	2.14 ± 1.56
MP-SPDZ hst. maj.	Preprocess (s)	0.01 ± 0.00	0.02 ± 0.01	0.03 ± 0.01	0.07 ± 0.09
	Online (s)	0.28 ± 0.05	0.49 ± 0.12	1.05 ± 0.30	4.32 ± 0.88
	External (s)	0.47 ± 0.05	1.33 ± 0.65	2.31 ± 0.95	7.02 ± 1.72
MP-SPDZ dishst. maj.	Preprocess (s)	35.98 ± 0.13	72.94 ± 2.19	123.68 ± 3.61	358.31 ± 11.08
	Online (s)	0.40 ± 0.01	1.77 ± 0.25	3.58 ± 0.95	13.39 ± 1.21
	External (s)	36.63 ± 0.13	75.59 ± 2.09	127.94 ± 3.45	373.59 ± 10.34
MOTION dishst. maj.	Preprocess (s)	0.37 ± 0.01	0.63 ± 0.08	0.70 ± 0.15	2.19 ± 1.99
	Online (s)	4.90 ± 1.14	5.91 ± 0.91	7.17 ± 1.56	10.43 ± 1.56
	External (s)	8.43 ± 1.57	9.53 ± 1.50	11.30 ± 2.22	15.69 ± 3.16

Protocols: Trio (HPMPC), Rep3/Semi2k (MP-SPDZ), BMR (MOTION)

Many improvements of MPC protocols have focused on moving parts of the computation to a separate preprocessing phase, which can be performed in advance without requiring any input data [4][70][71]. With this separation, one could set up a permanent connection between the participating nodes that prepare and store preprocessed material, such as Beaver Triples, until the next computation is performed. This reduces execution time through more efficient online calculations once the actual input data is available. Although several frameworks and protocols incorporate the idea, only Carbynestack has so far implemented an actual storage and exchange system for preprocessed material [72].

Here, we investigate the impact on runtime of separating preprocessing and online phases. Three of the four considered frameworks provide detailed information on the runtime of each phase, while MPyC does not support a separate preprocessing phase. However, as previously observed, the internal runtime measurements deviate from the external runtimes due to delays in setup and also between the frameworks, as they measure different aspects of the execution. Nevertheless, we investigate the internal preprocessing runtimes, especially in comparison to the online runtimes, as they still provide insight into the ratio between the two phases. The online runtime here refers to the time required to perform the actual MPC calculation using the previously generated preprocessed material and the input data. Whereas MOTION and MP-SPDZ always utilize a separate preprocessing and online phase, HPMPC provides a compile option to either separate the two phases or interleave them. The separated version always leads to higher end-to-end runtimes as the interleaved version efficiently uses the preprocessed materials once they are available. However, if the preprocessed material is already present before the execution, the external runtime would only include the online phase and thus benefit from this preprocessing model. In an optimal network, we observe that the runtime for the preprocessing phase only represents less than 20% of the external runtime for the MP-SPDZ honest majority protocol and MOTION, as shown in Table A.11. However, for the dishonest majority protocol in MP-SPDZ, the preprocessing phase accounts for more than 90% of the total runtime. This result has large consequences on the potential usability of dishonest majority frameworks in a fully developed system with asynchronous preprocessing material generation. For HPMPC, a similar effect is observed, where the preprocessing time accounts for roughly 30% of the external runtime. However, comparing the online runtime to the runtime with HPMPC's interleaved execution model when the separate preprocessing phase is turned off, the improvement is smaller.

Increased round-trip latency does not significantly alter the relative runtime difference between the preprocessing and online phases for the MOTION or MP-SPDZ honest majority protocol. For the MP-SPDZ dishonest majority protocol, we observe that while the difference is not as severe for higher round-trip latencies, an asynchronous preprocessing phase can still lead to runtime savings of over 20 minutes in a network with a 75 ms round-trip latency.

The effect of different bandwidths on the runtimes is evident in Table A.9. It shows that honest majority protocols benefit only slightly from a separate preprocessing phase. For MOTION, the effect increases with lower bandwidths. For a bandwidth of 100 Mbit/s, the preprocessing phase accounts for around 20% compared to only 5% for high bandwidth settings. For MP-SPDZ, the runtime of the dishonest majority protocol can be reduced tremendously by utilizing preprocessed materials. This effect is present for all bandwidths as the runtime increase for the online phase is significantly smaller than for the preprocessing phase.

Table A.10 shows the effect of different packet loss rates on the runtime. For the MOTION framework, both the preprocessing and online phases grow relatively similarly, with the online phase remaining the dominant factor in the overall runtime. For HPMPC, we observe that for a packet loss rate of 5%, the preprocessing phase accounts for approximately 25% of the external runtime. For MP-SPDZ, the results

differ between the honest and dishonest majority protocols. The increase in runtime for the preprocessing phase of the honest majority protocol is significantly smaller than that for the online phase, which further reduces its impact on the overall runtime. For the dishonest majority protocol, we see a relatively similar increase for both preprocessing time and online time. Therefore, the MP-SPDZ dishonest protocol benefits significantly from using preprocessed materials in all network settings.

Appendix A.2. Four-Server Secure Multiparty Computation

Although many protocols are specifically designed for a specific number of participating servers, all frameworks provide support for at least three- and four-party computation. As most use cases focus on a three-server setting, we focus on this scenario in our benchmark. In the following paragraphs, we also analyze the effect of adding an additional server on the runtime and the RAM usage based on the PSI workload. Both MP-SPDZ and HPMPC implement a state-of-the-art protocol secure against an active adversary in the four-party setting, since the additional server eases this implementation [8].

Table A.11 shows that the increase in runtime in optimal network settings of an actively secure 4PC protocol compared to a passively secure 3PC protocol is below 25% for all frameworks. HPMPC remains the fastest framework for the honest majority setting, while MOTION outperforms MP-SPDZ's dishonest majority protocol. Overall, adding an additional computation server introduces only a small runtime overhead compared to the large impact that different network parameters have on the MPC runtime.

Looking at the influence of network parameters, Table A.11 shows that the influence of the bandwidth is fairly similar in a three- and four-node setting. For HPMPC, the four-party variant even slightly outperforms the three-party variant, most likely because the framework can better distribute the required communication across all connections. Similar results are obtained for increasing latency, shown

Table A.11: Runtime in seconds for different bandwidths for three and four server Private Set Intersection (100 values each) [± indicates standard Deviation]. Honest Majority 4PC protocols are secure against an active adversary.

Framework	#Nodes	Bandwidth (Mbit/s)			
		100	1000	10000	25000
HPMPC	3	0.16 ± 0.01	0.03 ± 0.01	0.02 ± 0.00	0.02 ± 0.00
hst. maj.	4	0.15 ± 0.01	0.03 ± 0.01	0.03 ± 0.01	0.03 ± 0.01
MPyC	3	7.72 ± 0.28	7.93 ± 0.25	7.59 ± 0.20	7.62 ± 0.28
hst. maj.	4	8.51 ± 0.36	8.23 ± 0.34	8.26 ± 0.49	8.56 ± 0.62
MP-SPDZ	3	0.16 ± 0.01	0.12 ± 0.00	0.11 ± 0.00	0.11 ± 0.01
hst. maj.	4	0.22 ± 0.01	0.15 ± 0.01	0.14 ± 0.01	0.15 ± 0.01
MP-SPDZ	3	47.18 ± 0.13	40.87 ± 0.17	40.25 ± 0.18	40.30 ± 0.14
dishst. maj.	4	61.99 ± 0.07	53.42 ± 0.12	52.73 ± 0.07	52.75 ± 0.08
MOTION	3	9.54 ± 1.46	8.71 ± 1.03	8.79 ± 1.29	8.43 ± 1.57
dishst. maj.	4	9.51 ± 1.35	9.61 ± 0.99	9.16 ± 0.83	8.88 ± 1.09

Protocols: Trio/Quad (HPMPC), Shamir (MpyC), Rep3/Rep4/Semi2k (MP-SPDZ), BMR (MOTION)

Table A.12: Runtime in seconds for different round-trip latencies for three and four nodes private set intersection (100 values each; * 10 values each) [± indicates standard deviation]. Honest Majority 4PC protocols are secure against an active adversary.

Framework	#Nodes	Latency (ms)			
		0	25	50	75
HPMPC	3	0.02 ± 0.00	0.79 ± 0.11	1.57 ± 0.18	2.36 ± 0.25
hst. maj.	4	0.03 ± 0.01	0.85 ± 0.10	1.64 ± 0.15	2.47 ± 0.15
MPyC	3	7.62 ± 0.28	26.13 ± 0.29	45.94 ± 0.46	66.46 ± 0.63
hst. maj.	4	8.56 ± 0.62	26.75 ± 0.43	46.62 ± 0.51	66.80 ± 0.62
MP-SPDZ	3	0.11 ± 0.01	2.96 ± 0.01	5.68 ± 0.02	8.43 ± 0.02
hst. maj.	4	0.15 ± 0.01	3.95 ± 0.01	7.59 ± 0.00	11.25 ± 0.00
MP-SPDZ	3	40.30 ± 0.14	506.05 ± 0.14	971.25 ± 0.15	1436.33 ± 0.10
dishst. maj.	4*	7.57 ± 0.03	91.41 ± 0.03	175.18 ± 0.02	259.55 ± 0.11
MOTION	3	8.43 ± 1.57	14.37 ± 0.93	23.44 ± 0.82	32.08 ± 0.74
dishst. maj.	4	8.88 ± 1.09	14.66 ± 0.95	23.43 ± 0.62	32.75 ± 0.88

Protocols: Trio/Quad (HPMPC), Shamir (MpyC), Rep3/Rep4/Semi2k (MP-SPDZ), BMR (MOTION)

in Table A.12, where the gap in runtime between 3PC and 4PC remains relatively consistent for all frameworks except MP-SPDZ. Here, we see a larger increase for the four-node setting.

For the different packet loss rates, Table A.13 shows that the influence of packet loss appears to be more severe for 4PC compared to 3PC for all frameworks. The strongest difference is observed for the MP-SPDZ honest majority protocol, where the runtime for the four-node protocol is almost three times higher than for the three-node protocol in a highly unreliable network, compared to only a 25% difference without packet loss.

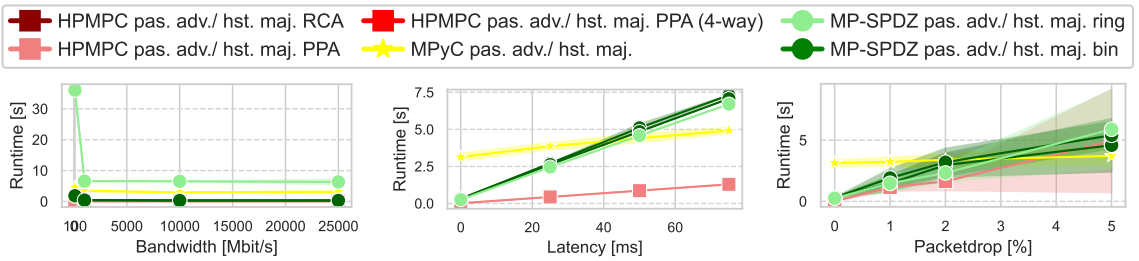
Table A.13: Runtime in seconds for different packet loss for three and four nodes private set intersection (100 values each) [± indicates standard deviation]. Honest Majority 4PC protocols are secure against an active adversary.

Framework	#Nodes	Packet Loss (%)			
		0	1	2	5
HPMPC	3	0.02 ± 0.00	0.23 ± 0.43	0.33 ± 0.49	0.83 ± 1.00
hst. maj.	4	0.03 ± 0.01	0.43 ± 0.52	0.66 ± 0.50	1.16 ± 0.53
MPyC	3	7.62 ± 0.28	9.30 ± 1.09	14.26 ± 0.80	33.28 ± 2.96
hst. maj.	4	8.56 ± 0.62	11.99 ± 1.39	16.85 ± 1.39	40.39 ± 1.91
MP-SPDZ	3	0.11 ± 0.01	0.94 ± 1.21	0.45 ± 0.49	2.07 ± 1.06
hst. maj.	4	0.15 ± 0.01	1.53 ± 0.90	2.55 ± 1.01	5.84 ± 1.59
MP-SPDZ	3	40.30 ± 0.14	83.08 ± 2.37	144.84 ± 5.09	419.45 ± 9.32
dishst. maj.	4	52.75 ± 0.08	145.41 ± 2.17	252.85 ± 4.52	698.37 ± 12.52
MOTION	3	8.43 ± 1.57	9.53 ± 1.50	11.30 ± 2.22	15.69 ± 3.16
dishst. maj.	4	8.88 ± 1.09	10.90 ± 1.00	12.73 ± 1.85	20.67 ± 2.55

Protocols: Trio/Quad (HPMPC), Shamir (MpyC), Rep3/Rep4/Semi2k (MP-SPDZ), BMR (MOTION)

Appendix B. Detailed Benchmarks

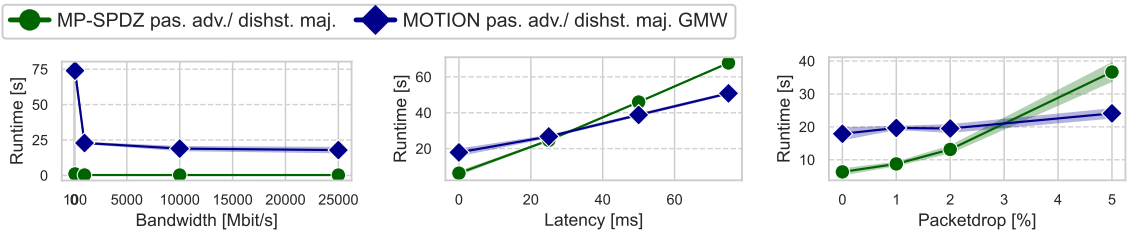
The following figures show the same measurements results as the ones presented in section 6 but with a linear scale and divided into honest and dishonest majority frameworks. This illustration helps to identify differences between frameworks in the same security class.



(a) Bandwidth (b) Latency (c) Packetloss

Figure B.10: Honest Majority: Multiplication benchmark in various network settings

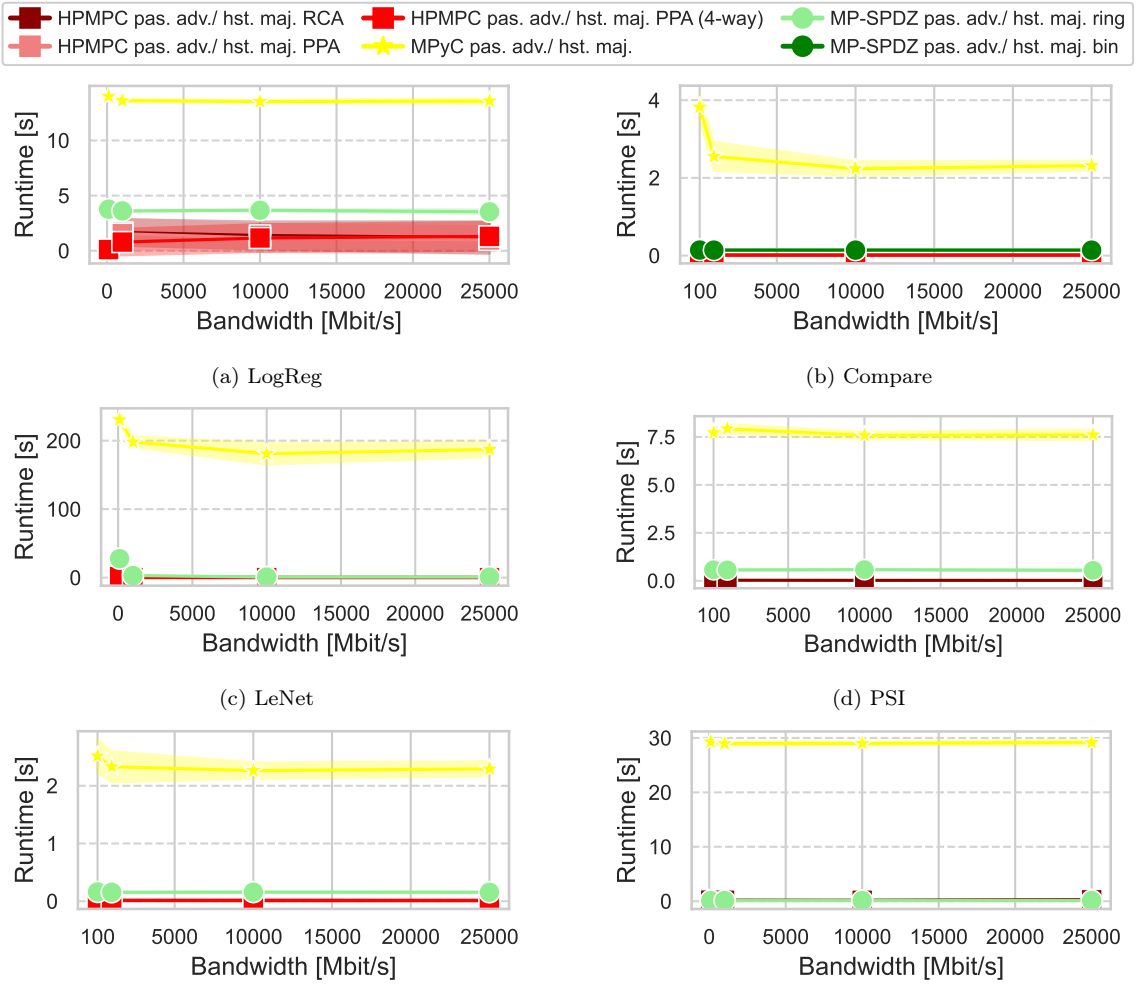
Protocols: Trio (HPMPC), Shamir (MPyC), Rep3 (MP-SPDZ)



(a) Bandwidth (b) Latency (c) Packetloss

Figure B.12: Dishonest Majority: Multiplication benchmark in various network settings

Protocols: Semi2k (MP-SPDZ), GMW/BMR (MOTION)



(a) LogReg

(b) Compare

(c) LeNet

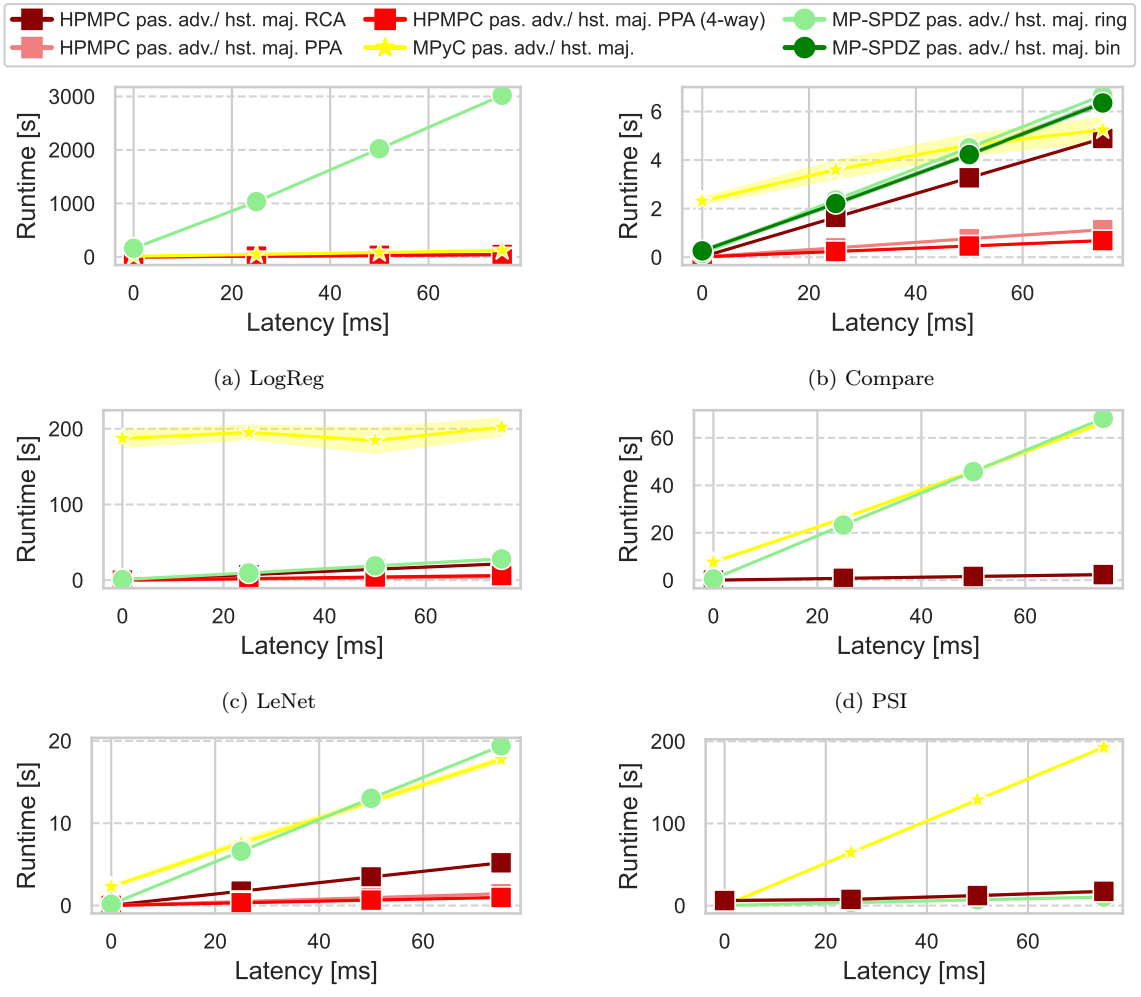
(d) PSI

(e) Auction

(f) AES

Figure B.14: Bandwidth comparison across different MPC frameworks and honest protocols

Protocols: Trio (HPMPC), Shamir (MPyC), Rep3 (MP-SPDZ)



(a) LogReg

(b) Compare

(c) LeNet

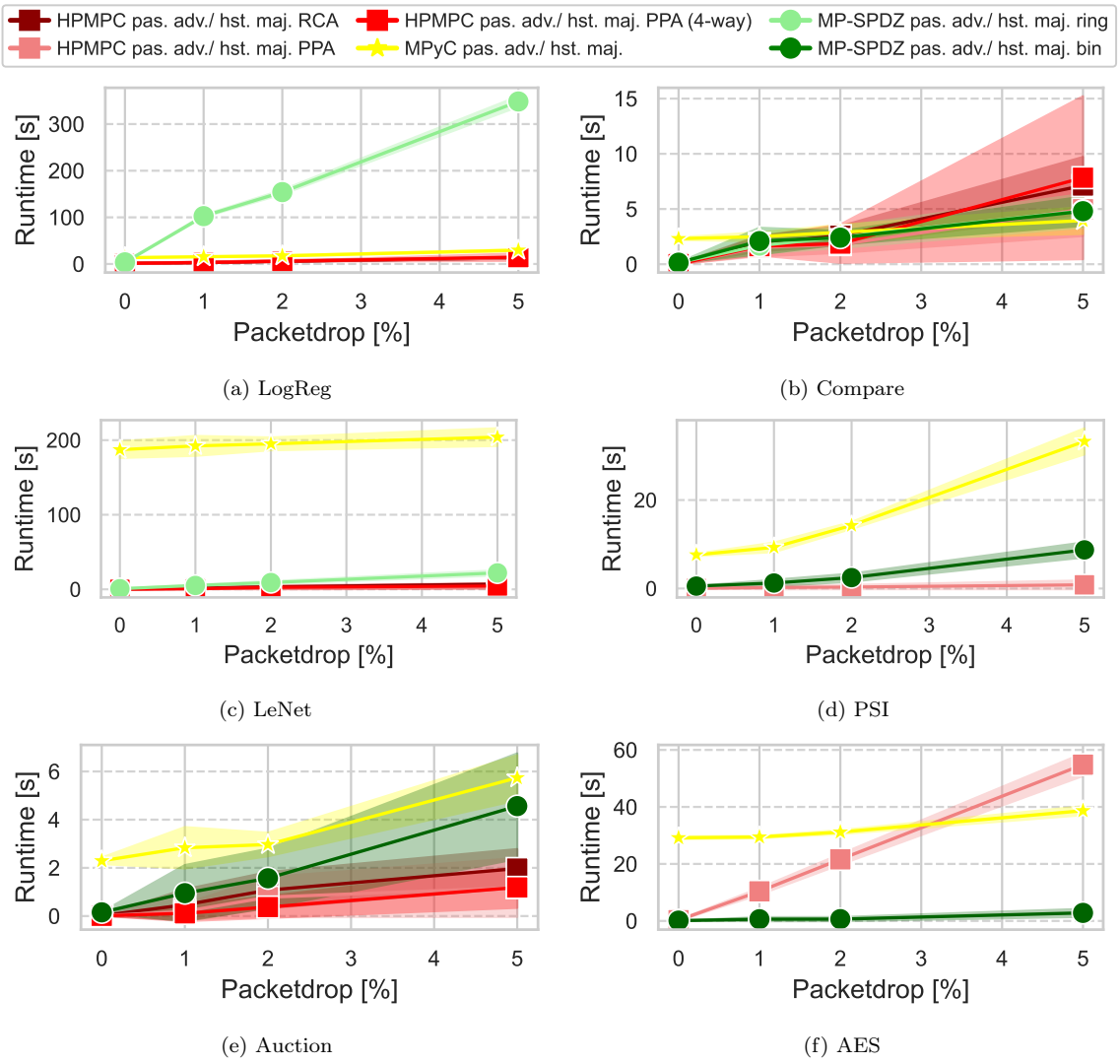
(d) PSI

(e) Auction

(f) AES

Figure B.16: Latency comparison across different MPC frameworks and honest protocols

Protocols: Trio (HPMPC), Shamir (MPyC), Rep3 (MP-SPDZ)



(a) LogReg

(b) Compare

(c) LeNet

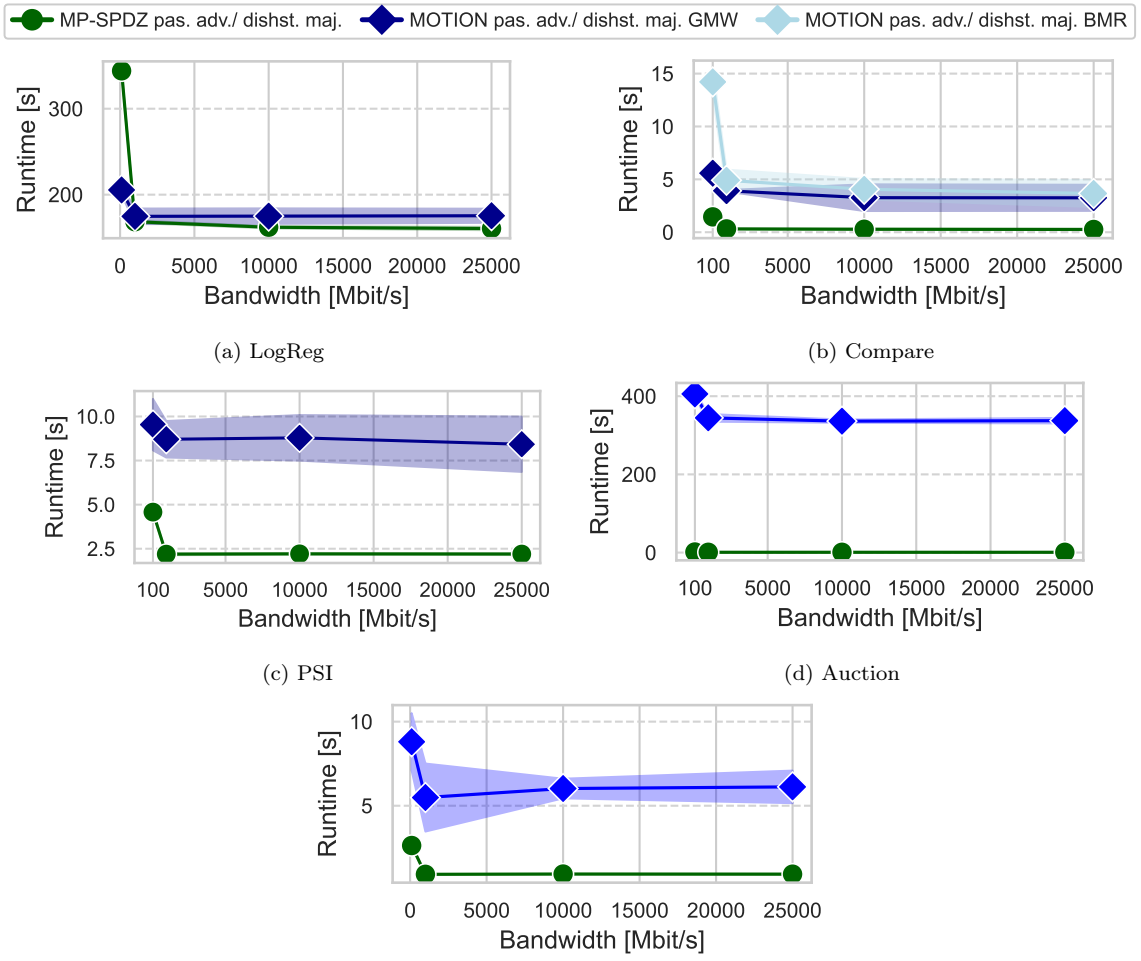
(d) PSI

(e) Auction

(f) AES

Figure B.18: Packetloss comparison across different MPC frameworks and honest protocols

Protocols: Trio (HPMPC), Shamir (MPyC), Rep3 (MP-SPDZ)



(a) LogReg

(b) Compare

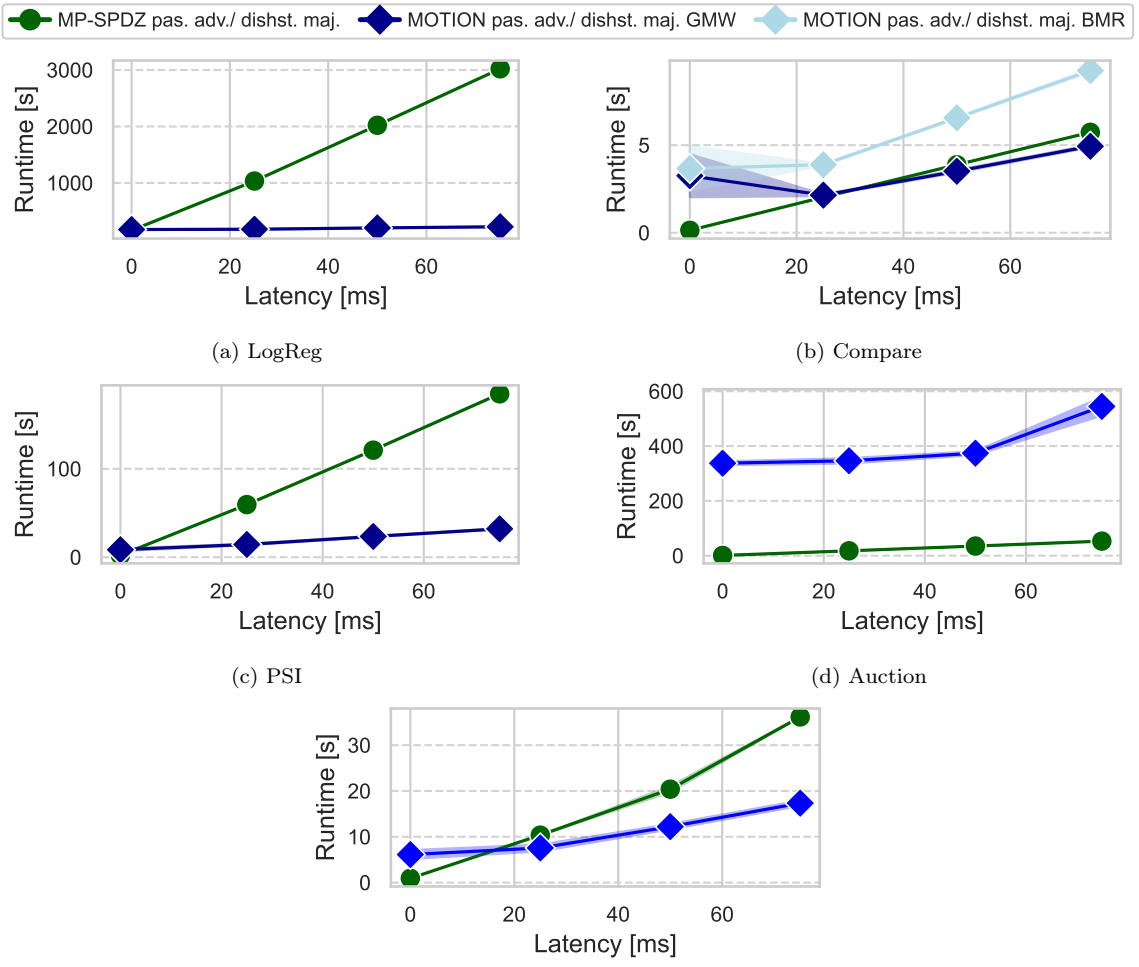
(c) PSI

(d) Auction

(e) AES

Figure B.20: Bandwidth comparison across different MPC frameworks and dishonest protocols

Protocols: Semi2k (MP-SPDZ), GMW/BMR (MOTION)



(a) LogReg

(b) Compare

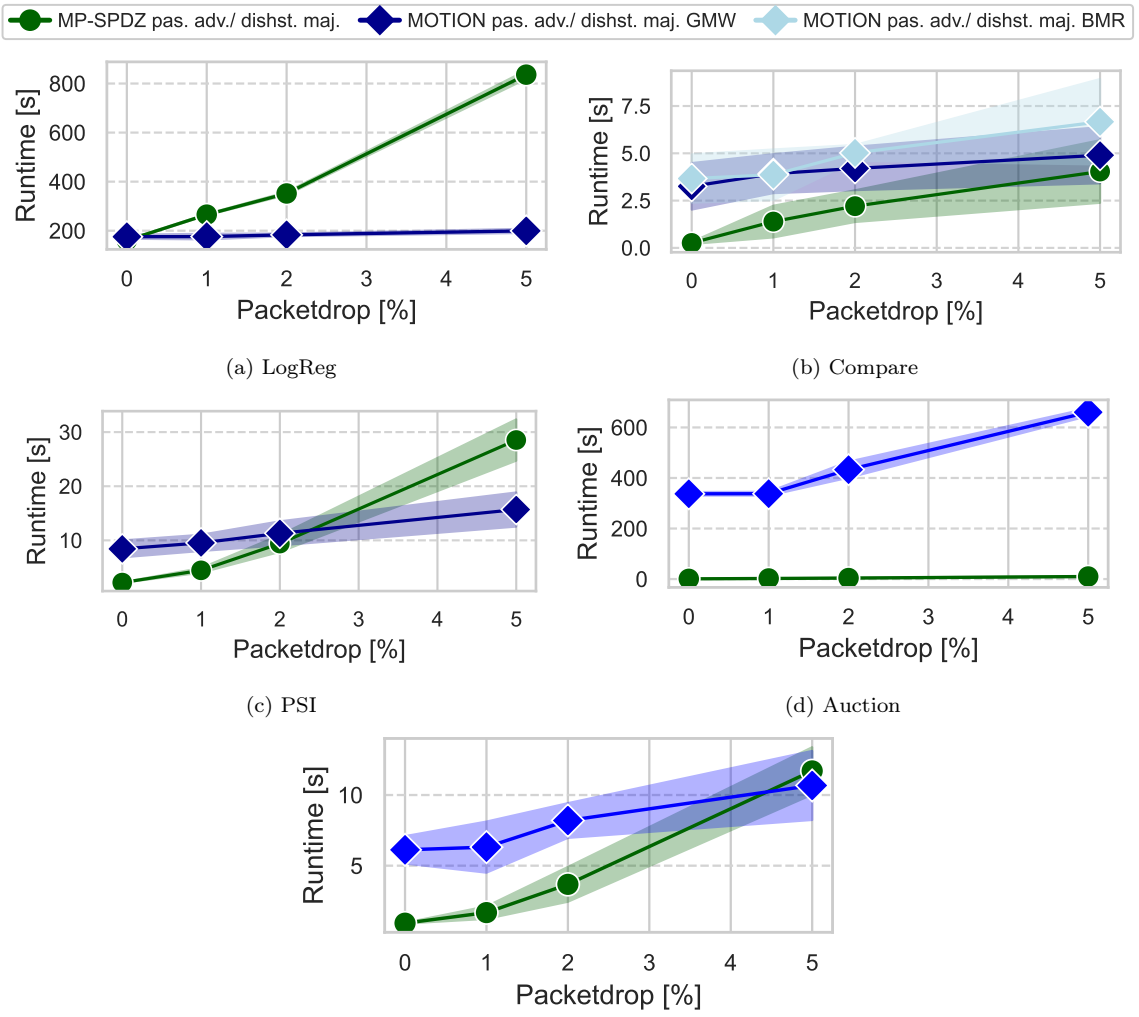
(c) PSI

(d) Auction

(e) AES

Figure B.22: Latency comparison across different MPC frameworks and dishonest protocols

Protocols: Semi2k (MP-SPDZ), GMW/BMR (MOTION)



(a) LogReg

(b) Compare

(c) PSI

(d) Auction

(e) AES

Figure B.24: Packetloss comparison across different MPC frameworks and dishonest protocols

Protocols: Semi2k (MP-SPDZ), GMW/BWR (MOTION)

Appendix C. Use Case Collection

Table C.14: Use Case Collection

Name	Description	Src.	clust.	Impl.	Sec. Mod.	Experiment Parameters	Hardware & Measured Performance
Coverage of a Data Set	In this use case a national statistics office wants to check the coverage of their dataset against a third-party source either to identify weaknesses in their collection method or to provide quality evaluation. This is done by performing a PSI between the shared identifiers of the two datasets.	[2]	PSI	PoC	Honest majority, passive adversary,	• 3 participants	—
Carbyne Stack	Carbynestack is an MPC framework based on MP-SPDZ with the goal of providing cloud integration through an easy-to-install cloud-based framework that can be deployed using standard cloud server tools. Additionally, the authors are currently integrating the Sharemind framework. One unique feature is the asynchronous generation of preprocessing material, enabling a true separation between the Offline and Online phases. Their first envisioned use case is a CO2 footprint calculation along a supply chain.	[72]	Secure Aggregation	PoC	—	• Arbitrary number of participants	• Works on any type of server hardware
Detecting Perpetrators of Sexual Misconduct	Callisto is an online platform for reporting sexual misconduct on college campuses and other educational institutions. They use MPC to securely match different entries, allowing them to form groups of victims. Studies have shown that victims of sexual misconduct are more likely to speak out publicly if they know other victims of the same perpetrator. Once a match is found through secure PSI, the data is forwarded to a lawyer who contacts the victims.	[47]	PSI	Impl.	—	• One server and local clients • Hash-based matching	—
Spam Filtering	In this Proof of concept, a spam detection algorithm is trained on emails distributed among different parties. This would enable different companies to train their detectors jointly without disclosing sensitive information in the correspondence. Since this is a relatively old POC, the algorithm used, namely Logistic Regression, is no longer a state-of-the-art approach to spam detection, but already achieves good results. To reduce computation time, they reduce the input size by preprocessing the text data and compressing the relevant information by dropping filler words.	[29]	Logistic Regression	PoC	Passive adversaries, dishonest majority	• 2 parties with one party sequentially changing for new input • CEAS Spam Filtering Dataset; 3,067 training and 200000 test documents;	• Measured runtime for offline and online phase, Training: 0.5s (data reduction) vs. 1.14 hours (original dataset); Testing: 5ms (cata reduction) vs. 4s (original data set). • Home computer; (3.2 GHz Intel Pentium 4 machine with 2 GB RAM and running 64-bit Ubuntu)

Table C.14: Use Case Collection

Name	Description	Src.	clust.	Impl.	Sec. Mod.	Experiment Parameters	Hardware & Measured Performance
Satellite Collision Detection	Many different countries and private companies deploy satellites into space, creating an increasing risk of satellite collisions. Although their rough position is publicly known, their exact trajectory is not. Using MPC to calculate whether their trajectories are likely to lead to a collision can save the participating parties huge sums of money by avoiding collisions through a small maneuver. It is, however, a challenging use case due to the high-precision floating-point calculations required in the collision estimation.	[39]	Numerical Analysis	PoC	Malicious security with abort, dishonest majority	• Two-party computation, Parties distributed over the globe • Small positional and movement parameters • Suggests the use of prepossessing in a real-world setup	• Number of Gates, runtime (Real, User, System Time) and Disk Usage for execution; Offline Phase (System Time): 27s $\pm$ 25s; Online Phase (System Time): 15s $\pm$ 10s • Local machine (virtual machine running 64-bit CentOS 6.4., 8 GB RAM, 4 processors, 100 GB disk space) • Avg. Calculation Error $7.027 * 10^{-5}$
Smart Meter	Smart Meters enable the precise tracking of resident energy consumption and enable a large set of opportunities for energy companies and public analysis. There is, however, a privacy issue in the deployment of smart meters: data from individual smart meters enables malicious actors to derive information from single households, such as regular absence. The Use Case proposes a flexible energy plan that allows for precise aggregation of energy consumption during peak and off-peak hours, which allows for separate individual pricing.	[43]	Secure Aggregation Secure Comparison	PoC	Passive adversary	• High number of participants in a specialized client-server structure	• Measured runtime • PoC is on a single machine with low performance; claim that it should be possible on smart meters
Patient Risk Stratification	A cooperation between multiple hospitals or medical data owners is envisioned to identify high-risk patients with multiple chronic diseases or an extremely high number of visits to medical institutions. This could enable doctors to quickly assess the medical situation and cater to the specific needs of high-risk patients.	[3]	PSI and Secure Statistics	PoC	Passive adversaries, dishonest majority	• Two-party protocol with iterative execution for multiple parties • Synthetic data set developed using an online platform that provides guidelines based on Census data; 2.7 million patients and 141 million encounters	• Measured runtime (a Few minutes for a smaller data set); Network Traffic (5.6 GB), and RAM Usage • Intel i7-8750H 2.2 GHz CPU and 16 GB memory • Simulated WAN environment with bandwidth 50 Mbit/s and 96 ms round-trip latency
Cryptographic Key Management	Sepior Key Management system enables organizations to deal with "multi-cloud, multikey and multiuser environments". The MPC component is threshold security, where the key is distributed among multiple parties that jointly compute signatures and decryptions. Here, the focus is on the AES algorithm.	[59]	Threshold Crypto	Yes	Passive and active security	• 3-5 parties per design	—

Table C.14: Use Case Collection

Name	Description	Src.	clust.	Impl.	Sec. Mod.	Experiment Parameters	Hardware & Measured Performance
Patient and Staff Tracking in Hospital	PoC with TNO Netherlands and Philips, where the goal is to track the movements of key staff, patients, and assets via real-time trackers and perform analysis on the data Personal Movement data is obviously highly private, so staff data only goes to the workers' council database, but MPC can be used to combine the different streams in order to increase efficiency and support the staff.	[49]	Private data analytics	PoC	—	—	—
Human Trafficking	Organizations such as Law Enforcement, NGOs, and Hospitals share information to monitor human trafficking. Through the exchange of their private data, they can identify networks of traffickers based on PSI or common routes.	[73]	PSI Private Aggregation Mean	PoC	—	• 3 participants	• On designated laptops connected over a WAN in the same region
Airline Slot Trading	Airlines are allocated designated departure slots, which are currently only traded within the same airline to prioritize expensive flights. The companies do not want to publicly trade these slots due to privacy concerns, as important information on their internal strategy could be disclosed. Using a private auction for these slots could facilitate fair price discovery and reduce the disclosure of sensitive information.	[74]	Secure Auction	PoC	—	• 3 participants	—
Financial Fraud Detection	Detect fraud using Google's page-rank algorithm in an MPC fashion. Intersections of multiple financial institution datasets are calculated to determine the flow of money, and then suspicious transactions are ranked based on their connectivity to known fraudulent accounts. They scale any fixed-point values to integers in terms of transfer amount to facilitate easier calculation.	[36]	Various	PoC	Passive adversaries, dishonest majority	• 3 participants	• Measured runtime (scales linearly with input size, 10 days in Python, 0.62 in C++), Bandwidth Requirement • 48 CPU cores (2.4 GHz Intel Xeon E5-2680v4) and 16 GB RAM, only locally tested • significant speed up when using multiple CPUs per party
Elderly Care	Multiple Organizations, such as hospitals, government/-municipalities, and caregiver,s share data to accurately calculate statistics without sharing PII This enables statistical insights such as spending on elderly patients with specific diseases, money saved due to specific legislation or care changes, and the average number of years elderly live in a ward. The use case description also contains information on a possible governance framework and management of cooperation	[75]	PSI Private Statistics on the Datasets	PoC	—	• 3 parties in the same region	—

Table C.14: Use Case Collection

Name	Description	Src.	clust.	Impl.	Sec. Mod.	Experiment Parameters	Hardware & Measured Performance
Mobility as a Service	Mobility as a Service through sharing and analyzing human movement data from different providers, such as local transport and mobile operators. The use case was proposed by the data sharing coalition in the Netherlands and is supposedly ongoing without current information	[76]	PSI and Statistical Computation	—	—	—	—
Sugar Beets	First real implementation of an MPC use case in a auction between sugar beet producers and the single sugar beet processor in Denmark to ensure a fair auction. The goal is to establish a market-clearing price where farmers input their offers, while the producer inputs the quantity they are willing to buy at a certain price. Implemented with a web surface where the farmers input their data, which is then secretly shared across three servers	[19]	Secure Auction	Yes	Passive adversary, honest majority	• 3 participating servers, >1000 clients in client-server model • 4000 bids per party	• Measured runtime (1.2 hours) • Laptops (Latitude D630) with 4GB RAM (Java being allocated 1500MB on each machine); Dual Core (2.2GHz) processor; running Windows XP Pro, and connected through an Ethernet LAN with 100 Mbit/s;
Student Dropout	IT-Industry in Estonia suspected that students of IT-subjects drop out more often than other students because they start working earlier and earn more money in student jobs. A joint analysis of data from the Ministry of Tax and Finance and the Ministry of Education was necessary, which is not possible in clear text. Therefore, MPC was used to compute the intersection and derive the analysis. One of the first use cases where data was preprocessed using MPC (e.g., attribute aggregation, row reduction)	[21]	PSI and Statistics Data Preprocessing	Yes	Passive adversary, honest majority	• 3 servers • 10 million tax records, 500.000 student records	• Measured runtime (384h WAN 60h local execution) • Local Laptop (2 Cores,2,4 GHz); Servers with 32 Cores, 8 GB RAM over WAN
Financial Performance Data Analysis	IT-Consortium sought to gain deeper insights into their business and HR strategy by sharing data between members, but no competitive information was to be disclosed. To the best of our knowledge, it was the first real-world use case with geographically separated nodes. The MPC task is to sort each column according to the values in order to hinder the gain of personal information before publication. Some other use cases, such as regression, were envisioned but not implemented.	[32]	Private Sorting	Yes	Passive adversary, honest majority	• 3 servers, >30 clients with client-server model	—
Boston Wage Gap	Calculating the Wage Gap between Men and Women in the Greater Boston Area is a goal of the Local Council. Local companies, however, are reluctant to participate if they have to send their statistics in the clear, as a large gap could lead to negative publicity. The data are therefore shared via MPC, and only the average gap is disclosed. Separate instances calculate the masked aggregate and the aggregate of the masking values, and then the service provider sends the sum while the analyzer subtracts mask	[77]	Aggr. and Averaging on the Dataset Secure Vector Addition	Yes	Passive adversary, dishonest majority	• 2 servers, 40-70 data providers	—

Table C.14: Use Case Collection

Name	Description	Src.	clust.	Impl.	Sec. Mod.	Experiment Parameters	Hardware & Measured Performance
CO_2 Allowance Trading	Airlines are required to buy certificates for their emissions in the EU emissions trading market, where the budget is allocated annually per airline and can be traded among them. By openly searching and selling CO2 allowances, the airlines reveal a great deal about their operations and internal business plans. Therefore, one option could be to do it privately using MPC.	[46]	Secure Auction	Conc.	—	—	—
Gene Testing	Genome Sequencing today is a common technology, and there is a large amount of genome data. Since it contains a lot of decodable information on the person, its privacy needs to be protected at all times. However, analysis of these datasets can provide important medical insights into specific biomarkers and their connections to diseases. This use case uses MPC to securely analyze MPC with a common Genome-Wide Association Study algorithm.	[78]	Secure Matching	PoC	Active and passive adversaries	• 2 servers • Several different standard datasets with between 10-20.000 samples	• Measured runtime (Up to 200 hours for 100.000k participants), Bandwidth Requirement, Required Storage Space • 3.47 GHz Intel Xeon X5690 CPU with 176 GB RAM • Observed Linear Growth of Execution time in the number of participants
Tax Fraud	Estonian companies have to transfer the VAT of their sold products to the state. The Ministry of Finance and Tax wants to use analytics to prevent tax fraud by comparing sales and purchases. These companies must disclose large sales and purchases, fearing the loss of sensitive information. Using MPC in this context could allow for tax fraud detection without disclosing private information.	[28]	PSI Aggregation and Comparison	PoC	Passive adversary, honest majority	• 3 participants • 20k to 80k companies providing sales reports to the ministry	• Measured runtime (38 minutes to 9 hours for different input sizes)
Advertisement Convergence	Google Ads and their customers both want to know their advertisement conversion rate, meaning how many people bought something after seeing the ad. Google has a set of views, while the advertisers have a set of purchases; they are intersected using common identifiers, and the sum of their purchases is aggregated. The focus of the implementation is on the monetary cost of algorithm execution, as Google has a large number of advertisers and views.	[79]	PSI (with Cardinality)	Yes	Passive adversary, honest majority	• Two servers • 100.000 elements a day, 1.000 protocol executions	• Measured runtime, Transferred Data used to calculate estimated cost on cloud instances • Google Cloud Instances • Compares three different possible algorithms with a focus on communication overhead

Table C.14: Use Case Collection

Name	Description	Src.	clust.	Impl.	Sec. Mod.	Experiment Parameters	Hardware & Measured Performance
SEPIA Network Event Aggregation	The Security through private information aggregation tool set by the ETH Zurich is specifically designed to allow for near real-time network event correlation and aggregation. The goal is to enable joint analyses of private network events, for example, in separated domains or even different companies, in order to detect anomalies or security breaches. Their specific protocol allows for quick comparisons by reducing communication overhead.	[37]	Compare and Aggregation	PoC	Passive adversary, honest majority	• 5-25 input parties and 3-9 computing servers	• Measurement of running time and data (Scales linearly with the number of servers and input) • 2x Pentium 4 CPU (3.2 GHz), 2 GB memory, and 100 Mb/s network • variation of peers and input nodes
Human Resource Modeling	Employee turnover and recruiting are major cost factors for companies. Usually, employee data is highly sensitive, so sharing it even within the company can be a complex process. Training retention and promotion models through Secure Multiparty Computation enabled the company in this use case to utilize 15% more data than before, resulting in improved models.	[80]	PSI, Logistic Regression	Yes	—	—	—
Fraud Detection in Financial Institutions	Banks lose over a billion dollars in financial fraud and already use sophisticated models to stop it. Today, they do, however, only use the data of their one bank since no exchange of information between institutions exists and is currently illegal. By exchanging the information in a secret shared form and training the detection model on more data, the quality increased significantly.	[81]	PSI, Logistic Regression	PoC	—	• 3 participants	—
Secure Survey	Early (2004) theoretical use case using MPC to conduct a private survey. The idea stems from the yearly Taulbee Survey that the Computer Science Faculties in the US conduct yearly to gain an overview of academic and financial trends for salaries and employment. Participation is usually low, and large security concerns are communicated over sharing income data. An MPC solution could reduce the leakage of private information and increase participation.	[82]	Secure Aggregation and Statistics	PoC	Passive adversary, honest majority	• 2 computation servers with 196 input parties	• Only theoretic insights (Number of Gates) • Major bottleneck to their implementation is the memory consumption required for the circuit
Confidential Benchmarking and Credit Rating	As farmers are often creditors with seasonal bad performance and high debt ratio, Danish banks try to better predict their credit scores and adapt their portfolio to their specific needs. Accounting data for farmers are very sparse but present at one consultancy house and linear programming is used to identify high-performing farmers. The MPC setup then consists of 2 servers one at the bank and one at the consultancy house and the data sets are preprocessed to only identify high-performing farmers.	[33]	PSI Secure Linear Programming	PoC	Dishonest majority, malicious adversary	• 2 servers • 2500 samples in one data set and 7500 in the other • Modified SPDZ protocol, implemented in FRESCO	• Measured runtime (25 seconds) • Amazon EC2 instance (m4.large with 8GB RAM and 2 cores, 2.4GHz Intel Xeon Processor)

Table C.14: Use Case Collection

Name	Description	Src.	clust.	Impl.	Sec. Mod.	Experiment Parameters	Hardware & Measured Performance
Clinical Trial Participant Selections	Clinical Trials often require patients with certain preconditions or illnesses to compare the effectiveness of their treatment. These data are stored in patient databases which are highly protected by data privacy laws such as HIPAA. This implementation uses MPC to access these data stored at different health service providers and first perform a secure comparison on numerical and textual data and even Logistic Regression to find the best fit.	[83]	Secure Comparison Logistic Regression	PoC	—	—	—
Privacy-Preserving Speed Advisory Systems	The goal of this use case is to establish a vehicle-to-vehicle MPC connection that is able to find a consensus speed in a motorcade scenario. The goal is to identify the speed at which cars on average produce the least CO2. As this speed depends on the specs and age of the vehicle, it is private information that through MPC could be kept private while allowing automated determination of ideal speed in a certain range.	[84]	Secure Aggregation and Comparison	PoC	Passive adversaries, dishonest majority	• 20 participants • 19 speed-emission pairs leading to 152 bytes shared • For 20 participants communication is less than 3KB	—
Jana system	The Jana system is a full database framework using MPC for all operations such as inputs, outputs, or operations on the stored data. The data is secretly shared between different servers during input and never combined. Implements different algorithms to enable different database operations. Additionally, it allows the selection of different protocols depending on the security level required.	[59]	Secure Querying of PSI connected databases	PoC	Flexible security model	—	—
Italian Bank Case	The bank of Italy and the Italian National Institute of Statistics (ISTAT) want to join their databases using the tax identifier. This join will enable more insight into people and allow for statistical analysis. In this first implementation, only an additive scheme is used which can be performed on the different columns.	[2]	PSI and Aggregation	PoC	Passive adversary, honest majority	• Synthetic data	—
Samsung Secure Aggregation	Goal is to link two different datasets horizontally based on a shared identifier. Since in this use case not only aggregate statistics but also full tables are supposed to be published, their first step is to perform a PSI and ID anonymization protocol in which the common identifiers are found and then changed to a random value. After this the databases can be joined, aggregate statistics can be calculated, or the full table can be disclosed.	[2]	PSI and Statistics	PoC	—	• 2 participants • Dataset from 1m to 20 million generated using Faker (Generator for Personal Data)	• Measured runtime (2 to 30 min in private LAN) • Laptops
Augmented Heart Disease Analysis	In this implementation different healthcare providers work together to train a more accurate prediction model for cardiovascular diseases. Using a larger sample size together with an increased feature set through PSI enables better early prediction.	[50]	PSI and Linear Regression	Impl.	—	—	—

Table C.14: Use Case Collection

Name	Description	Src.	clust.	Impl.	Sec. Mod.	Experiment Parameters	Hardware & Measured Performance
Statistics Korea	Statistics Korea are developing a Data Hub for exchange, correlation, and aggregation of data using PSI , general MPC, and homomorphic protocols, as well as Synthetic Data Generation for statistics analysis and Logistic Regression. The goal for the platform would be the linkage of all governmental records in order to enable large-scale data insights.	[2]	PSI and Logistic Regression	Pilot	—	• 8576 franchises for Korean food and their store space, id, turnover, etc.	—
US Student Analysis	First prototype in the US on federal government level as part of a push towards privacy-enhanced data sharing. Proof of Concept was implemented by linking the data between student debt and the National Student Aid Study Group to reproduce statistical insights on Attendance, Loan Balance, and Grant Amount for students in both databases.	[85]	PSI and Statistical Analysis	Pilot	Passive adversary, honest majority	• 2 participants • 123 experiment experiments	• Measured System time (4.8 hours of CPU time) • Two cloud-based virtual machines
International Trade Analysis	International Trade is subject to many different privacy laws, but joint aggregate statistics could provide valuable insights for business and law enforcement. This Concept Study focuses on analyzing trade between different European Countries to mitigate illegal or tax-avoiding transfer of money and goods over borders.	[24]	PSI and Aggregation and report creation	Pilot	—	• 5 participants in different countries	—
Statistics Netherlands E-Health	Goal is to improve a health care service, namely an E-Coaching App, by combining insurance data with hospital data in private matters. This is part of a 10-year cooperation plan that analyzes the effect of innovation in comparison to the cost. Analysis of effects is only possible by combining app usage data with patient medical data through MPC for privacy reasons. The concrete example was patients with inflammatory bowel disease and the effect of their use of a health coach app.	[86]	PSI with Statistics	Pilot	Passive adversary	• 3 participants distributed over one small country • 4350 samples (Synthetic Data for PoC and Real data for Pilot) • Mixture of Shamir Secret Sharing and Pallier Homomorphic Encryption	• Deployed on Servers of different companies
Carrier heart disease	The project focused on establishing a legal framework for the reuse of patient data, specifically for heart diseases. The data collected previously from different sources were reused to calculate statistical insights into risk indicators and treatment successes. For this, several PET technologies were used.	[87]	PSI and extensive calculation	Concept	Passive adversary, honest majority	• 4 participants	—
Heart Failure Prediction	PoC with several Dutch hospitals cooperating to train a linear regression model based on a horizontally split data set that predicts the risk of chronic heart failure in patients within the next year.	[49]	Lasso regression after PSI	PoC	—	• Patient data up to 10,000 samples in synthetic data set	• Measured runtime (between 1 and 30 minutes for different feature sizes)

Table C.14: Use Case Collection

Name	Description	Src.	clust.	Impl.	Sec. Mod.	Experiment Parameters	Hardware & Measured Performance
Crypto Wallet Protection	Threshold security allows secret sharing of a cryptographic private key, which is used for signing cryptocurrency transactions by the wallets. By sharing the key, no single point of failure exists increasing the security of the wallet. Several different companies implement the technology for their wallets using several different algorithms.	[88]	Digital signature using shared private key	Yes	Dishonest majority	• Arbitrary number of participants, tested for 3	• Measured runtime (2.1 seconds) • Standard Laptops • Almost no latency
Machine Learning Inference for fMRI datasets	The authors present a full pipeline for secure inference of fRMI scans on brain activity. Since these scans already allow the derivation of information on the general health, emotional state, and even character traits of a patient, they are highly sensitive. This implementation shows that through MPC it is possible to obtain classifying information through private inference using a pre-trained model hosted on an external server without disclosing private information	[53]	Private Inference	PoC	Honest majority, passive adversary	• 2 participants	—
Drug Target Interaction Prediction	The prediction of drug/target interaction requires large-scale data sets often derived from expensive experiments and therefore immensely valuable. Nonetheless, the model's accuracy might profit from increased input data through a collaboration of multiple entities. The authors prove that this is achievable in an acceptable amount of time using MPC. The training of their Neural Network takes approximately 4 days on 1 million features.	[54]	Neural Network Training and Inference	PoC	Dishonest majority, passive adversary	• over 1 million training samples	• Measured time (4 days) and accuracy
Protecting Symmetric Keys	The Kerberos Authentication Service in Windows uses AES and the hashed user password to encrypt the session key. Therefore each breach allows one to impersonate a user. If the keys are stored on two different servers and MPC is used to calculate the symmetric encryption, this risk is greatly minimized. This is proposed and tested by this use case.	[69]	AES with shared key	PoC	Passive adversary, dishonest majority	• 3 participants • 33 parallel AES blocks	• Measured throughput (35.000 logins per second, 1.3 million AES executions per second) • 10-core Intel Xeon (E5-2650 v3) processors and 128GB RAM, 10 GBit/s bandwidth • Latency of 232ms for a login

Table C.14: Use Case Collection

Name	Description	Src.	clust.	Impl.	Sec. Mod.	Experiment Parameters	Hardware & Measured Performance
Inventory Matching	JP Morgan wants to enable inter-client trade of stocks without leakage of bids and offers. Currently, banks offer intra-bank trading to clients to allow them to save costs on additional stock exchange fees. Traditionally, clients would have to disclose their buying and selling intents to the bank to enable the matching, but with MPC this can be done privately. This is requested by companies since they worry that the bank could use their data to its advantage. In the use case, several different approaches and security models were tested.	[44]	Find matches based on secure comparison between quantity	Yes, live in company	Active adversary, honest majority	• Two or three servers, 100 - 10000 clients • PoC, range of 100 to 10000 samples • Browser-based client interface	• Measured time (10.4s per transaction), Received and Send Message Size • Local PoC Intel i7, 6 cores, 2.6GHZ and i5,4cores 2GHz; servers AWS type c5a.8xlarge with 32 vCPUs • Linear runtime increase for more samples
Podiatrist Data Exchange	The Dutch Association of Podiatrists, as well as the government of the Netherlands, see a growing need for research in the area of podiatrists as treatment costs have increased in recent years. Through the collaborative use of patient and treatment data, they want to identify best practices as well as predict future illnesses based on patient data	[35]	Secure Statistics and Prediction	Yes	—	—	—
Biometric Identification	Early Proof of Concept Using the Eigenface algorithm for Face Recognition on Private Input Compared to a Public Database. This could be used in the search for criminals or for identification at a location without enabling the tracking of movements or registration since only the result of the admittance request and not the identity will be public.	[89]	Eigenface algorithm	PoC	Passive adversary, dishonest majority	• 2 participants • One image • C++ implementation of proprietary algorithm based on Pallier Encryption Scheme	• Measured accuracy (96% lower due to quantization) and Time (40s)
Internet Access Points Routing	Routing Information and Connections need to be exchanged but can disclose private information. The current BGP protocol is not an ideal solution in terms of convergence and overhead, but switching to a "centralized MPC solution" would allow for more efficient routers, better security, and guaranteed convergence in the search for optimal routing tables. This is what is proposed in this use case.	[90]	Secure information exchange and routing table construction	PoC	Passive adversary, honest majority	• K computation servers; client-server model	• Computation time only estimated 0.13 seconds for 19 neighbors F

Table C.14: Use Case Collection

Name	Description	Src.	clust.	Impl.	Sec. Mod.	Experiment Parameters	Hardware & Measured Performance
Dark Pool Trading	Dark Pools in stock exchange trading have risen in popularity as traders do not want to disclose large orders due to successive price increases. Therefore at the moment, stock brokers offer so-called black pools for offers which are then fulfilled if successively but firms mishandle the data regularly leading to over 200 Million $ in fines in the last years. JP Morgan aims to provide an MPC solution that allows for a dark pool with privacy protection and without manipulation where orders are automatically matched. In the proof of concept, multiple approaches are tested including a bucket model, a volume model with arbitrary order size, and a model with multiple buckets and iterative clearing.	[45]	Secure matching of arbitrary number of sell and buy bids	PoC	Passive adversary, honest majority	• N=[10,100,1000,10000] input parties; Client-server model with three servers	• Measured runtime (between a few ms and 2s for different loads) • Single CPU without communication
Demand Response for Private Energy Trading	Private producers and consumers of energy want to provide their flexibility in demand and supply to the grid operators. The surplus and device planning information is highly private as a lot of information can be derived from the energy consumption of a household. Furthermore, it does not make sense for grid operators to purchase at the household level. The use case therefore proposes the implementation of a secure aggregation of the offers, then a double auction in which these offers are met with the bids of the grid operators, and then a linear programming phase in which the device planning calendars are derived.	[91]	Secure aggregation Double auction and then opened to perform linear programming	PoC	Honest majority, passive adversary	• 3 participants, client-server model	• Measured runtime (30s) • Intel i-9900 CPU and 128GB of RAM • 1ms ping time between devices
MPC and Smart Contracts for Data Market Place	This proof of concept proposes to combine MPC with Smart Contracts to enable secure selling of data in a marketplace with individual data providers and buyers. Both parties agree on a smart contract that includes a reference to the MPC code which will be executed on a separate MPC engine before the results are published to the buyer. This allows for fair payment for the data and secures privacy since the smart contract cannot be changed once committed.	[92]	Secure computation of arbitrary algorithms on encrypted data	PoC	Honest majority, passive adversary	• 3 participants	—
Gboard	Federated Training of the keyboard on Google devices using secure aggregation of the recommendations and typing patterns over all keyboards. An elaborate training model based on the smartphone state and participation model is deployed. MPC can be used since only the average weights of words and their combinations are required.	[41]	Secure Aggregation	Yes	Honest majority, passive adversary	• One server, 100-500 clients • 100k Entries with 62 Bits	• Measured time and communication per party (1 MB)
Predictive Maintenance for distributed fleets	Inpher and Defense Department of US Distributed Calculation on current vehicle data to improve maintenance, private inference of model based on health of device	[93]	Secure Statistics and Prediction	Yes	—	—	—

Table C.14: Use Case Collection

Name	Description	Src.	clust.	Impl.	Sec. Mod.	Experiment Parameters	Hardware & Measured Performance
Advertisement Analysis	Allow the cross-website analysis of advertisement campaigns through PSI of browser and client reports with the same ID. This data can then be used for simple reporting or even for training predictive machine learning models.	[94]	PSI and Statistics	PoC	Passive adversary	• 3 participants • Synthetic Data (1.000.000. bits)	• Measured Cost for Network and Computation in Cloud • 3 m5dn.12xlarge instances
Liquidity Matching	Banks these days use central systems provided by national banks to solve issues when large-scale transactions should be processed immediately. This requires larger liquidity of the banks as the funds are transferred electronically in real time. As large transactions often exceed the liquidity a system is required to obliviously sort and execute incoming and outgoing transactions to balance liquidity.	[95]	Specific Algorithm	PoC	Active adversary, Honest majority	• 3 participants, up to 1000 clients • Artificial Transactions (10 to 100)	• Measured Online and Offline runtime (0.5s to 50s for different variants) • Intel i-9900 CPU, 128GB of RAM
MOZAIK	The authors propose an end-to-end key data collection, analysis, and sharing system. One key feature is the use of MPC which allows using secret shared symmetric and asymmetric keys, for data encryption and communication.	[96]	(A)symm. encryption, Machine Learning	PoC	Active adversary	—	—
Home Loan Calculation	Singapore's housing market is largely controlled by the government and public housing. To rent an apartment from the government, certain factors need to be fulfilled such as a positive debt ratio and the ability to make a down payment. The proposed PoC allows the calculation of the loans terms without disclosing information to other parties.	[27]	Secure Statistics	PoC	Passive adversary; dishonest majority	• 3 participants • Up to 10.000 values	• Measured runtime (scales linear with input) • Windows 10 Home with Intel Core i7-4720HQ Processor, 16GB RAM, and JVM heap size 2GB
Human Activity Recognition	More and more research is conducted on measuring human activity via sensors to learn more about general health patterns and physical reactions to certain situations. This data is often analyzed using machine learning models. The others propose to use MPC to secure the proprietary model and the private user data.	[30]	Neural Network Training and Inference	PoC	Passive adversary; honest and dishonest majority	• 2 or 3 participants • 7500 samples and 45 features	z • Measured "acceptable" accuracy and runtime
Fairness in Neural Networks	Oasis and Meta partner to perform secure and private analysis of Meta's neural networks based on gender and race. They use MPC to perform a private survey on the users and to privately share the model weights for the customer which is then analyzed based on the survey results to assess whether large differences for the races and genders exist.	[97]	Secure Survey and Statistics	Yes	—	—	—

Table C.14: Use Case Collection

Name	Description	Src.	clust.	Impl.	Sec. Mod.	Experiment Parameters	Hardware & Measured Performance
HIV Treatment	Doctors only have a very limited amount of time to choose the right medicine for HIV patients. The goal is to find the right suppressor that prohibits the outbreak of AIDS the longest. Both patient data and the doctor's choice are, however, private. They propose a long-term secret shared database that allows them to derive the most suitable choice from previous decisions.	[98]	Secure Statistics	PoC	—	• 2 participants • Artificial dataset with up to 20000 samples	• Measured runtime (Scales linearly with input size)
Text Classification	The authors propose and implement a Naive Bayes Classifier for text recognition with a focus on SMS fraud detection. The MPC solution could enable better detection together with higher privacy standards.	[99]	Naive Bayes Classifier	PoC	Passive adversary; honest majority	• 2 participants • One SMS with 5200 words or 369 words • implemented in Rust	• Measured runtime (21ms for small sms) • c5.9x-large EC2 instances with 36 vCPUs, 72.0 GiB of memory and 32 threads, 1 GBit/s
Robot Collision Detection	In many future scenarios, autonomous vehicles, drones, or robots will have to interact in a distrusting environment, for example in traffic or when different companies need to cooperate. Here MPC can help to allow for collision prevention and the setup of meeting points for basic interaction. The authors propose several algorithms that allow for this functionality based on Garbled Circuits.	[100]	Collision Detection and Prevention Algorithms	PoC	Active adversary, dishonest majority	• 2 participants • Small positional and velocity information	• Measured accuracy and feasibility • Laptops with Ubuntu 12.0 SO, Intel Atom at 2.0 GHz, and 2 GB RAM
Kidney Exchange Problem	Patients requiring a kidney transplant often try to find willing donors among their friends and family. However, more often than not, these donors are not medically compatible. To use this willingness and to increase the chances of finding a willing donor, many countries have created a central registry that stores incompatible patient, and donor pairs and matches them with other pairs. The authors propose to use secure multiparty computation for these central registries to prevent potential loss of viable patient information.	[101]	Integer Programming	PoC	Passive adversary, honest majority	• 3 servers, multiple clients • Real data, 2900 patient donor pairs • Used MP-SPDZ for implementation	• Measured runtime for different Latency (from a few seconds to half an hour) • AMD EPYC 7702P 64-core processor (single core assigned to each virtual server)
Cardiogram Classification	Analyzing Cardiogram data can be efficiently done using local models, however, similar to all machine learning models they could profit from more data. The authors propose a private inference on a model provided by a third party that has a highly accurate model. MPC is used to protect private patient data during the inference.	[102]	Neural Network Classifier	PoC	Passive adversary, dishonest majority	• 2 participants	• Measured theoretical Complexity and runtime (17s) • Intel Core Duo, 3.0 GHz, 4 GB of RAM

Table C.14: Use Case Collection

Name	Description	Src.	clust.	Impl.	Sec. Mod.	Experiment Parameters	Hardware & Measured Performance
Lightning Network IoT Bitcoin Payment	IoT devices participating in the Lightning Network (LN) secretly share private keys and commitment points with the LN gateway to protect against collusion and ransom attacks.	[31]	Threshold Cryptography	PoC	Passive adversary, dishonest majority	• 2 participants • ECDSA 2 Key shares	• Measured runtime and Required Bandwidth (sufficient for real-time payments) • Raspberry Pi and Intel Xeon Scalable processor,8 GB RAM
Portfolio Analysis	Banks offer a service to buy a complete portfolio from stockholders. However, to obtain an offer currently only a summary of the stocks without specifics is sent to the bank to protect the privacy of the broker. This leads to inaccurate pricing by the bank potentially harming either the bank or the broker. The authors propose to perform the pricing algorithm using 2-Party Secure Multiparty Computation which allows the bank to have full insight into the portfolio without loss of privacy.	[103]	Secure Statistics	PoC	Passive adversary, dishonest majority	• 2 participants • Small values (stock pricing, volume, etc)	• Measured runtime and calculated number of gates (feasible, runtimes below 30 seconds)