Universiteit Leiden

Privacy-Preserving Collaborative Vulnerability Assessment using Secure Multi-Party Computation

Rafael Cordova Niewold (s3993744)

Supervisor:
Eleftheria Makri

Bachelor Thesis– Data Science and Artificial Intelligence

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

July 1, 2026

Abstract

Modern organizations increasingly depend on shared digital infrastructures, cloud services, software supply chains, and external service providers. This creates interdependence between such systems and introduces the need for collaborative cybersecurity assessment. This can be deduced, as vulnerabilities or risks observed within one organization may affect subsequent systems. However, directly sharing vulnerability reports, patching delays, risk scores, or threat indicators can expose sensitive information about an organization's security posture. Hence, this thesis investigates how collaborative vulnerability or risk assessment can be designed and implemented using Secure Multi-Party Computation (MPC), while preserving private organizational inputs and evaluating computational and communication performance.

In order to investigate this, a proof-of-concept was developed using MP-SPDZ. To compute functions using the platform, synthetic organizational security data was generated to represent vulnerability exposure indicators. These include factors such as a CVSS-like severity score, number of critical vulnerabilities, patch delay, and asset exposure. The implemented MPC functions compute group-level metrics such as an average weighted exposure score, a threshold count, a group alert, and a maximum exposure score, while revealing only the final outputs.

Following experiments with different input values, the results show that the implementation works correctly for multiple numbers of simulated organizations. Runs with $n = 3$, $n = 5$, and $n = 10$ completed with runtimes below 0.11 seconds, while larger runs with $n = 25$ and $n = 50$ increased to 1.17 seconds and 5.95 seconds, respectively. The same trend was visible in the communication measurements, where global communication increased from less than 5 MB for $n = 10$ to 62.22 MB for $n = 25$ and 474.63 MB for $n = 50$. Ultimately, the results indicate that MPC can support privacy-preserving collaborative vulnerability assessment in a simplified proof-of-concept setting, but that scalability and protocol choice remain important limitations for larger deployments.

Contents

1 Introduction

Modern organizations rarely operate in isolation. Through systems such as cloud services, software supply chains, and third-party service providers, an ever increasing interconnected digital infrastructure arises. Among others, this suggests a major security flaw: cascading vulnerabilities. A breach or cybersecurity risk in one system can lead to various impacts on companies that rely on its systems or services. This is especially relevant for sectors in which data is highly sensitive. As example, cloud computing is increasingly discussed as an important technology for healthcare data processing and storage [1]. However, this imposes security and privacy risks, as data now also relies on the safety of connected systems.

This interdependence motivates means of cybersecurity through collaboration. Companies, especially those protecting sensitive data, contain a wide variety of security data and methods. Hence, collaboration in this context suggests combining information from multiple interconnected companies. In this manner, they may obtain a more accurate view of shared vulnerabilities, common threats, or systemic cyber risks. For instance, companies may desire information regarding the number of participants that observe similar vulnerabilities, or whether a common threat indicator has been observed by several parties. This proves certain merit to the solution at hand. As enforced by previous works, the practice of sharing information in pursuit of improved cybersecurity may lead to benefits such as proactive detection and improved situational awareness [16]. More specifically, the paper argues that cyber security information sharing supports collective defense by enabling use cases such as discovering covert attacks and new malware, issuing early warnings, distributing threat intelligence, and providing advice on how to secure networks. This is relevant for collaborative vulnerability and risk assessment because individual organizations may only observe a partial view of the broader threat landscape.

However, collaboration through information sharing also introduces an important limitation: the direct sharing of cybersecurity data is often problematic. Vulnerability reports, patching delays, internal risk scores, and threat intelligence may reveal sensitive information about an organization's security posture and structure. If such data is disclosed, it could lead to reputational damage, regulatory concerns, competitive disadvantages, or even new attack opportunities [16]. Therefore, organizations require methods that allow them to benefit from collaborative security analysis without directly exposing their private inputs.

Secure Multi-Party Computation (MPC) provides a possible solution to this problem. MPC allows multiple parties to jointly compute a function over their private inputs without revealing those inputs to one another [20, 12]. In the context of this thesis, this means that several organizations could jointly compute a vulnerability or risk metric, such as an average risk score or a threshold-based alert, while keeping their individual security data private. However, privacy-preserving computation also introduces computational and communication overhead. Therefore, it is necessary to study not only whether such a system can be designed, but also whether it can be implemented with acceptable performance.

1.1 Problem Statement

The central problem addressed in this thesis is that organizations may benefit from collaborative vulnerability or risk assessment, but cannot safely share their raw security data because it may expose sensitive information about their internal security posture. The challenge is therefore to

determine whether Secure Multi-Party Computation can be used to compute useful group-level vulnerability assessment metrics while preserving the privacy of each participant's input data and maintaining acceptable runtime and communication performance.

1.2 Research Question

The preceding discussion shows that collaborative cybersecurity can provide valuable insights, but that direct sharing of vulnerability or risk data is often impractical due to privacy and confidentiality concerns. Secure Multi-Party Computation offers a possible way to address this tension, but its usefulness in this setting depends on which assessment functions are computed, how the participating organizations are modeled, and whether the resulting system has acceptable computational and communication performance. Therefore, this thesis investigates the following research question:

> How can collaborative vulnerability or risk assessment across multiple organizations be designed and implemented using Secure Multi-Party Computation, while preserving the privacy of each participant's security data and maintaining acceptable computational and communication performance?

1.3 Research Sub-questions

In order to navigate this research question, this study will attempt to build a proof-of-concept MPC implementation by adopting previous studies and existing implementation frameworks. Using this, functions will be defined and created to be computed on synthetic datasets. Hence, some major research sub-questions arise to further traverse the landscape introduced by the research question:

1. What types of collaborative vulnerability or risk assessment functions must be defined to be suitable for privacy-preserving computation?
2. What system model and threat model are appropriate for collaborative vulnerability or risk assessment using MPC?
3. How can selected assessment functions be implemented using an MPC framework such as MP-SPDZ?
4. What are the runtime, communication, scalability, and correctness characteristics of the resulting proof-of-concept implementation?

Through addressing these questions, this thesis will attempt to provide an accurate response to the larger query at hand, and in doing so aid the cybersecurity space.

1.4 Contributions

This bachelor thesis makes three main contributions. First, it provides a structured overview of related work on privacy-preserving cybersecurity collaboration, including private set intersection, secure aggregation, and MPC-based approaches. Second, it defines a system model for collaborative vulnerability or risk assessment in which organizations compute joint metrics without revealing their private inputs. Third, it implements and evaluates a proof-of-concept using synthetic input

data and an MPC framework, focusing on correctness, runtime, communication cost, and scalability. This bachelor thesis was conducted at the Leiden Institute of Advanced Computer Science (LIACS) under the supervision of Eleftheria Makri.

1.5 Thesis Overview

This thesis is structured as follows. Section 2 introduces the necessary background on collaborative cybersecurity, vulnerability assessment, cyber risk assessment, and secure multi-party computation. Section 3 discusses related work on privacy-preserving cybersecurity collaboration. Section 4 describes the proposed system model, threat model, input data and synthetic dataset design, collaborative assessment functions, and evaluation metrics. Section 5 presents the proof-of-concept implementation in MP-SPDZ, including the development environment, synthetic data generator, and various aspects of carrying out the experiment. Subsequently, the Section 6 reports the experimental results. Finally, Section 7 discusses the findings, trade-offs, limitations, and implications. Section 8 concludes the thesis by answering the research question and providing insight into future research.

2 Background

The background section introduces the major points of understanding and knowledge necessary for further topics. Consisting of five subsections, it starts with collaborative cybersecurity, which will elaborate on the aforementioned concept of information sharing. Following this, vulnerability and cyber risk assessment will be covered. These sections elucidate what vulnerabilities are and how they are assessed, as well as the differences between vulnerabilities and risks. Lastly, it introduces Secure Multi-Party Computation as the privacy-preserving computation technique used in this thesis, followed by a brief description of MP-SPDZ as the implementation framework.

2.1 Collaborative Cybersecurity

Collaborative cybersecurity refers to security practices in which multiple organizations exchange or process security-relevant information between one another in order to improve their defensive capabilities [16]. This form of collaboration can be carried out in various ways. From an operational perspective, information sharing may include indicators of compromise, such as malicious IP addresses, file hashes, or domain names. On the other hand, when analyzing more analytically, companies may exchange information surrounding attack patterns, vulnerabilities, incidents, or risk indicators. In this case, the latter proves more relevant to this thesis, as the various indicators can be represented as structured, measurable inputs. This makes them more suitable for privacy preserving collaboration, as multiple organizations can jointly compute aggregated security metrics, by using MPC functions on numerical data. More generally, NIST SP 800-150 describes cyber threat information sharing as a structured activity in which organizations define sharing goals, identify relevant sources, scope sharing activities, and establish rules for publishing and using threat information [9].

A key distinction is that not all shared cybersecurity information has the same level of abstraction. Technical threat intelligence often consists of low-level and time-sensitive indicators, while vulnerability and risk information may describe broader properties of an organization's security

posture. Tounsi and Rais [18] emphasize that technical threat intelligence must be understood in relation to its type, usefulness, lifetime, and operational context. This matters for collaborative assessment because the type of information being processed determines what kind of joint computation is meaningful. For example, identifying common indicators of compromise is a set-based problem, whereas computing an average risk score or threshold alert requires numerical aggregation. Hence, when defining the types of functions carried out by the MPC, such key distinctions must be established.

2.2 Vulnerability Assessment

A vulnerability, in the case of technology and cybersecurity, can be defined as a flaw, weakness, or misconfiguration within a system. Such weaknesses may be exploited by an adversary, and can cause varying levels of harm to organizations. Hence, vulnerability assessment is the process of identifying such weaknesses, evaluating their severity, and prioritizing them for remediation. The process of such assessment often involves vulnerability scanning, patch management, manual analysis, and the use of vulnerability databases.

In the context of this thesis, the most relevant aspect of vulnerability assessment is that vulnerabilities can often be represented through structured and measurable indicators. A Common Vulnerability Scoring System (CVSS), is often used. A CVSS provides a numerical score that reflects the severity of a vulnerability and can be translated into qualitative categories such as low, medium, high, and critical [6]. Such scores do not capture the entire security posture of an organization, but they provide a useful abstraction for comparing and prioritizing vulnerabilities.

In addition to severity scoring, vulnerability assessment also commonly involves prioritization and remediation decisions. In practice, not all vulnerabilities can be patched immediately, which means organizations must decide which vulnerabilities require attention first. NIST SP 800-40 describes enterprise patch management as preventive maintenance for technology, helping organizations reduce the likelihood of compromises, data breaches, and operational disruption [17]. Similarly, CISA's Known Exploited Vulnerabilities catalog is intended as input for vulnerability management prioritization, especially when vulnerabilities are known to be actively exploited [4]. This is relevant to the proof-of-concept, because the synthetic input variables include both vulnerability severity and patch delay.

This abstraction is important for privacy-preserving collaboration. Instead of requiring organizations to reveal full vulnerability reports or internal system details, a collaborative system may operate on derived indicators. As such, it may include average vulnerability severity, the number of critical vulnerabilities, or the number of unresolved vulnerabilities above a certain threshold. These values are far easier to model as private numerical inputs in an MPC protocol. Hence, in this thesis, vulnerability assessment is not treated as a full scanning or auditing process, but as a source of structured input values that can be used for collaborative computation.

2.3 Cyber Risk Assessment

Cyber risk assessment extends vulnerability assessment by considering not only the existence or severity of weaknesses, but also their possible consequences. In general, risk assessment looks at two major aspects of a vulnerability: likelihood and impact. Specifically, NIST SP 800-30 describes risk assessments as part of a broader risk management process, intended to provide decision-makers

with information needed to determine appropriate responses to identified risks [8]. NIST SP 800-37 further places risk assessment within a broader risk management framework, in which organizations manage security and privacy risk through structured steps such as categorization, control selection, assessment, authorization, and continuous monitoring [5].

The distinction between vulnerability and risk is important for this thesis. As established, a vulnerability score provides various beneficial metrics necessary to cybersecurity. However, it may not necessarily determine the actual risk for an organization. For example, a vulnerability that affects a non-critical internal system may be less risky than a vulnerability affecting a public-facing system which stores sensitive data. Therefore, risk assessment often analyzes factors such as asset value, exposure, likelihood of exploitation, patch delay, and potential business impact.

From the perspective of MPC, cyber risk assessment is relevant because these factors can be expressed as input variables in a joint computation. For example, organizations may privately provide numerical indicators such as vulnerability severity, exposure level, estimated likelihood, or impact score. A collaborative risk function could then compute an aggregate risk score, a maximum risk value, or a threshold-based alert without revealing each participant's individual values. This makes cyber risk assessment a suitable setting for investigating whether MPC can support privacy-preserving collaborative security analysis with acceptable performance.

2.4 Secure Multi-Party Computation

Secure Multi-Party Computation (MPC) is a cryptographic technique that allows multiple parties to jointly compute a function over their private inputs, without revealing those inputs to one another. The concept was introduced by Yao [20] and has since developed into a broad area of cryptographic research. In general terms, MPC separates the computation of a result from the disclosure of the underlying data. Instead of requiring parties to send their raw data to a central entity, the parties participate in a protocol that reveals only the agreed-upon output.

This is directly relevant to the problem addressed in this thesis. When looking at collaborative vulnerability or risk assessment, it can be observed that each organization may hold sensitive input values such as patching delays, vulnerability scores, or risk indicators. Such values, due to their sensitivity, are prone to reputational damage or exploitation, as they would expose information about an organization's security posture. However, if the relevant assessment function can be expressed mathematically, MPC can be used to compute the result while keeping the individual inputs private. For example, several organizations could compute an average risk score, the maximum observed vulnerability score, or whether a certain number of participants exceed a risk threshold, without disclosing each organization's individual values.

An important aspect of MPC is the threat model under which the protocol is evaluated. A common distinction is between semi-honest adversaries, who follow the protocol but try to learn additional information, and malicious adversaries, who may actively deviate from the protocol. Lindell [12] explains that the security guarantees of MPC protocols must always be interpreted in relation to such assumptions. For this thesis, the precise threat model will be defined in the methodology section, since it determines what kind of privacy guarantees the proof-of-concept aims to provide.

While MPC offers strong privacy benefits, it also introduces additional computational and communication costs. Computations that are simple in a non-private setting may require multiple rounds of interaction between parties in an MPC setting. Therefore, the usefulness of MPC for

collaborative vulnerability or risk assessment cannot be assessed only by whether the computation is theoretically possible. It must also be evaluated in terms of correctness, runtime, communication overhead, and scalability. These trade-offs form a central part of the evaluation in this thesis.

2.5 Shamir Secret Sharing

Shamir secret sharing is a cryptographic method in which a private value is divided into multiple mathematical shares [15]. Instead of one party holding the full secret directly, each party receives only a share of the secret. A sufficient number of shares can be used together to reconstruct the original value, while an insufficient number of shares does not reveal the secret.

This idea is important for MPC because it allows computations to be performed without revealing the original input values. In a secret-sharing-based MPC protocol, parties compute on shares rather than on the raw private values themselves. For example, if an organization provides a private vulnerability score, that score can be split into shares and distributed across the participating parties. The parties can then jointly compute functions over these shares, such as sums, averages, comparisons, or threshold checks, without directly seeing the original score.

In the context of this thesis, Shamir secret sharing is relevant because the final implementation uses the Shamir protocol in MP-SPDZ. The private input values of each organization are treated as secret values, and the computations are carried out over shared representations of those values. Only the final group-level outputs are revealed. This matches the privacy-preserving goal of the proof-of-concept, since individual organizational inputs and intermediate scores remain hidden during the computation.

2.6 MP-SPDZ

MP-SPDZ is an open-source framework for implementing and benchmarking Secure Multi-Party Computation protocols. Proving relevant to this thesis, it allows for access to a high-level programming interface. The platform allows for writing computations, which are then executed using varying MPC protocols and security settings. Keller [10] describes MP-SPDZ as a versatile framework that supports a wide range of MPC protocol variants under different assumptions, including honest-majority and dishonest-majority settings. The official repository also describes MP-SPDZ as software for benchmarking MPC protocols across several security models and underlying technologies, including secret sharing, homomorphic encryption, and garbled circuits.

For this thesis, MP-SPDZ is relevant because it allows the proof-of-concept to focus on the assessment functions rather than on implementing cryptographic protocols from scratch. In practice, this means that collaborative vulnerability or risk assessment functions can be written as MPC programs, while MP-SPDZ handles the underlying secure execution. The thesis can therefore investigate how selected functions behave when executed privately, and compare this to their non-private equivalents.

MP-SPDZ supports different MPC protocols with different security assumptions and performance characteristics [10]. This is relevant because the choice of protocol can influence both security guarantees and practical performance. For example, MASCOT is a malicious-secure arithmetic MPC protocol based on oblivious transfer [11], while Shamir-based protocols rely on secret sharing as introduced by Shamir [15]. In this thesis, these protocol differences are not compared experimentally,

but they are relevant for understanding why protocol choice may affect scalability and communication cost.

The use of MP-SPDZ also supports the experimental part of this thesis. As previously stated, the goal is to evaluate the feasibility and performance of such MPC systems. Hence, the framework can be used to measure how selected computations behave under different input sizes, or amount of simulated participants. This makes it possible to study whether functions such as aggregate risk scoring or threshold-based alerts remain practical when computed in a privacy-preserving manner. In this way, MP-SPDZ serves as the implementation platform connecting the theoretical idea of MPC to the experimental evaluation of collaborative vulnerability and risk assessment.

3 Related Work

This section discusses previous work related to privacy-preserving cybersecurity collaboration. Whereas the background section introduced the main concepts used throughout the thesis, this section focuses on existing approaches that attempt to enable collaboration without fully revealing sensitive security data. It first considers privacy-preserving threat intelligence sharing, followed by private set intersection and secure aggregation techniques. It then discusses MPC-based privacy-preserving analytics more broadly, before positioning this thesis in relation to the existing literature.

3.1 Privacy-Preserving Threat Intelligence Sharing

Privacy-preserving threat intelligence is a method which limits the disclosure of sensitive organizational data, whilst aiming to preserve the benefits of collaborative cybersecurity. It attempts to do so through typically exchanging information such as indicators of compromise, attack patterns, malware observations, or general vulnerability-related information. As discussed by Skopik et al. [16], such sharing can support collective defense by improving situational awareness, enabling early warnings, and helping organizations respond more effectively to emerging threats.

However, direct sharing of threat intelligence can expose information that organizations may prefer to keep confidential. For example, sharing observed indicators of compromise may reveal what an organization monitors, which attacks it has encountered, or which systems may have been affected. This creates a tension between the value of collaboration and the sensitivity of the underlying data. As a result, several privacy-preserving approaches focus on allowing organizations to correlate or aggregate threat intelligence without fully revealing their private observations.

Existing privacy-preserving approaches in this area often focus on specific collaborative tasks. For example, organizations may want to determine whether they have observed the same malicious IP address, file hash, or domain name, without revealing their complete internal logs. Alternatively, they may want to compute how frequently a particular threat has been observed across a group of participants without revealing each participant's individual contribution. These tasks motivate the use of technical building blocks such as private set intersection and secure aggregation, which are discussed in the next subsection.

3.2 Private Set Intersection and Secure Aggregation

Private Set Intersection (PSI) is a cryptographic technique that allows two or more parties to compute the intersection of their private sets without revealing the non-overlapping elements. This allows for organizations to identify specific indicators of compromise that occur between participants, while still keeping all other indicators private. This proves useful when organizations want to identify common threats without exposing their full monitoring data. Over-threshold variants of PSI extend this idea by identifying elements that appear in at least a certain number of participants' datasets, which can be useful for collaborative intrusion detection or prioritizing indicators observed by multiple organizations [13, 2].

Secure aggregation is rather similar in nature, as it attempts to secure inputs between participants. However, instead of identifying overlapping set elements, secure aggregation takes a different approach, by computing aggregate statistics. Computations may include sums or counts, while hiding each participant's individual input. In cyber threat intelligence, this can be used to compute the number of sightings of a specific threat indicator without revealing which organization contributed how many sightings. Veugen et al. [19] demonstrate such an approach for cyber threat intelligence, where aggregation results are only published when enough private inputs are present. Practical initiatives such as SACTI further illustrate how secure aggregation can support privacy-preserving publication of threat intelligence statistics [14].

These approaches are highly relevant to privacy-preserving cybersecurity collaboration, but they also illustrate certain limitations that are important for this thesis. PSI is primarily suited to set-based questions, such as determining whether organizations share common indicators. Secure aggregation is well suited to simple statistical outputs, such as totals or counts. However, collaborative vulnerability and risk assessment may require more general numerical functions, such as weighted risk scores, maximum values, or threshold-based alerts over multiple input variables. This motivates the use of more general MPC techniques, which can express a broader range of computations.

3.3 MPC-Based Privacy-Preserving Analytics

Secure Multi-Party Computation provides a more general framework for privacy-preserving analytics than task-specific methods such as PSI or secure aggregation. Ideally, an MPC structure may allow parties to compute arbitrary functions over private inputs. This may occur as long as the function can be represented in a form supported by the underlying protocol. This makes MPC suitable for settings where the desired computation goes beyond set intersection or simple aggregation.

Practical MPC frameworks have made it easier to implement and evaluate such computations. Sharemind, for example, demonstrates how secret-sharing-based MPC can be used as a framework for privacy-preserving computation in practical settings [3]. MP-SPDZ provides a more recent and versatile framework for implementing and benchmarking a wide range of MPC protocols [10]. Such frameworks are important, as they allow researchers to focus on the design of privacy-preserving applications and performance evaluation, rather than implementing cryptographic protocols from scratch.

For this thesis, MPC-based analytics are relevant because vulnerability and risk assessment can be modeled as computations over structured numerical inputs. Instead of asking whether two organizations share the same indicator, the system may compute an average vulnerability score, a

maximum risk value, or whether a group-level threshold has been exceeded. Such functions are naturally expressed as numerical computations, making them suitable for implementation in a general MPC framework. At the same time, MPC introduces computational and communication overhead, which means that feasibility must be evaluated experimentally.

3.4 Positioning of This Thesis

The aforementioned works show that privacy-preserving cybersecurity collaboration has been studied mainly through methods such as threat intelligence sharing, private set operations, and secure aggregation. These approaches, while proving foundational for our understanding of protecting sensitive organizational data during collaboration, are still rather limited in nature. This is mainly due to their emphasis on carrying out specific tasks, such as finding shared indicators or aggregating sightings.

However, they are still relevant to this thesis, due to their foundational nature. Hence, this thesis attempts to build on these ideas, and apply them to collaborative vulnerability and risk assessment. Rather than focusing only on whether organizations have observed the same threat indicator, the thesis investigates how organizations can jointly compute assessment metrics derived from vulnerability or risk indicators. This includes functions such as aggregated scores, maximum values, or threshold-based alerts. The goal is not to develop a full production-ready cybersecurity platform, but to design and evaluate a proof-of-concept that demonstrates how such computations can be performed using MPC.

The positioning of this thesis can therefore be divided into three major parts. First, it connects privacy-preserving collaboration techniques to the specific setting of vulnerability and risk assessment. Second, it defines concrete assessment functions over synthetic organizational security indicators. Third, it implements these functions using an MPC framework and evaluates correctness, runtime, communication overhead, and scalability. This positioning allows the thesis to assess both the privacy-preserving potential and the practical performance costs of MPC in collaborative security assessment.

4 Methodology

Within the following section, the methodology used to investigate privacy-preserving collaborative vulnerability and risk assessment will be defined. The goal of the approach is to define a simplified and measurable proof-of-concept, as opposed to building a full production-ready cybersecurity platform. Through this, multiple organizations will be able to jointly compute vulnerability exposure metrics without revealing individual inputs. The methodology consists of a variety of subtopics, including defining a system model, threat model, input representation, collaborative assessment functions, MPC implementation strategy, and evaluation metrics.

4.1 System Model

The system model, in this case, consists of multiple organizations that participate in a collaborative vulnerability exposure assessment. Within the MPC protocol, an organization is represented as a party. For the initial proof-of-concept implementation, three parties will be used, corresponding

to three participating organizations. For said parties, each holds private security-related input values, which will be used for calculation. These input values describe their exposure to a specific vulnerability-related risk scenario.

The chosen scenario is collaborative assessment of exposure to unpatched critical vulnerabilities. This scenario was selected because it can be represented using structured numerical indicators, while still remaining close to realistic cybersecurity practice. Hence, each organization may have information about the severity of vulnerabilities, the number of critical vulnerabilities, patching delays, and exposure of affected assets. In a real setting, such values may be obtained in various ways such as vulnerability scanners, asset inventories, or internal risk management systems. In this thesis, they are represented using synthetic integer inputs.

The system computes only group-level outputs. Individual organizational inputs and intermediate scores are not revealed. Instead, the parties jointly compute values such as an average exposure score, the number of organizations above a threshold, or a group-level alert. This reflects the intended use case of collaborative assessment: organizations obtain information about the collective risk situation without directly disclosing their own security posture.

4.2 Threat Model

The proof-of-concept assumes a semi-honest threat model. Semi-honest refers to participating organizations following the protocol as specified, but may attempt to infer additional information from the messages they observe or from the final output. This assumption is commonly used in MPC research as a first step for evaluating privacy-preserving computation, because it allows the thesis to focus on whether the collaborative assessment functions can be computed correctly and efficiently.

Under this threat model, each organization is allowed to know its own input values and the final output of the computation. However, such organizations are under no circumstance able to learn the private input values of other participating organizations. For example, if an output demonstrates that three participants exceed a vulnerability exposure threshold, it may not reveal who, nor what their exact scored were.

Under this model, the thesis does not consider malicious adversaries that deviate from the protocol, submit false inputs, or attempt to disrupt the computation. These attacks are outside the scope of the proof-of-concept. The focus is instead on demonstrating the feasibility and performance of privacy-preserving collaborative assessment under clearly defined assumptions.

4.3 Input Data and Synthetic Dataset Design

The input data in this thesis is designed around the requirements of the collaborative assessment functions. The proof-of-concept uses synthetic data with realistic ranges, rather than real organizational security data. This is appropriate because real vulnerability and risk data are rather difficult to obtain, as they would themselves be sensitive and raise ethical considerations.

Each organization provides four private integer inputs:

- `cvss_score`: a CVSS-like vulnerability severity score scaled from 0 to 100;
- `critical_vulnerabilities`: the number of unpatched critical vulnerabilities;

- `patch_delay`: the average patch delay in days;
- `asset_exposure`: an exposure score representing whether affected assets are public-facing or business-critical, scaled from 0 to 100.

The four input variables were selected because they capture different but complementary aspects of vulnerability exposure. The `cvss_score` represents the technical severity of the vulnerability, while `critical_vulnerabilities` captures how many unresolved severe issues are present. The `patch_delay` variable adds a time-based remediation aspect, since vulnerabilities that remain unpatched for longer may represent a higher exposure. Finally, `asset_exposure` represents the context of the affected assets, since vulnerabilities in public-facing or business-critical systems may be more relevant than vulnerabilities in isolated or less important systems.

These four variables were chosen instead of a smaller set because using only severity or vulnerability counts would ignore remediation time and asset context. At the same time, the model was kept limited to four variables instead of adding many additional factors, such as exploit availability, business impact, compensating controls, or sector-specific requirements. This keeps the proof-of-concept simple enough to implement and evaluate in MPC, while still allowing the assessment function to combine multiple meaningful dimensions of vulnerability exposure.

The CVSS-like score is scaled to avoid decimal input values. For example, a CVSS score of 8.0 is represented as 80. This is mainly done because each organization's private input values are provided to MP-SPDZ as integers in the input files. Hence, the weighted exposure score can be computed using integer-based operations over secret values. Although MP-SPDZ is able to support other numeric representations, using scaled integer inputs keeps the proof-of-concept simpler and makes the MPC outputs easier to compare with the plaintext expected results. The synthetic inputs are not intended to fully model all aspects of organizational cyber risk. Instead, they provide a controlled and measurable representation of vulnerability exposure that can be used to evaluate the MPC proof-of-concept.

4.4 Collaborative Assessment Functions

The proof-of-concept defines several collaborative assessment functions. These functions are designed to represent different levels of computational complexity and different types of group-level information.

Initially, a rather simple average exposure score is computed. Within this function, each participating organization may provide one private exposure score. Using this, the MPC protocol will simply compute an average across all organizations. Although rather simple in nature, it verifies the basic MPC pipeline: private inputs are provided, a joint computation is performed, and only the eventual output will be revealed to all participants.

Second, a weighted vulnerability exposure score is computed. For each organization i, the score is defined as:

$$score_i = 4 \cdot cvss_i + 2 \cdot critical_i + 2 \cdot patch_i + 2 \cdot exposure_i$$

where $cvss_i$ is the scaled vulnerability severity score, $critical_i$ is the number of critical vulnerabilities, $patch_i$ is the patch delay, and $exposure_i$ is the asset exposure score. In this case, $cvss_i$

has a higher weight as vulnerability severity is the central indicator, whereas the other values are rather additional contributing factors. The group-level weighted average is then computed as:

$$average = \frac{1}{n} \sum_{i=1}^{n} score_i$$

This function represents a simplified vulnerability exposure metric that combines multiple security indicators into one numerical value.

Third, a threshold count function is defined. This function counts how many organizations have a weighted exposure score above a predefined threshold T:

$$count = |\{i : score_i \geq T\}|$$

This function is useful because it reveals how many organizations are above a high-exposure threshold without revealing which organizations exceed it.

Fourth, a group alert function is defined. This function outputs 1 if at least k organizations exceed the threshold, and 0 otherwise. In this case, k is treated as a public protocol parameter. This means that the participating organizations agree beforehand on how many high-exposure organizations are required to trigger the alert. The value of k itself is therefore not secret. Instead, the private information lies in each organization's input values and in whether each individual organization exceeds the threshold. The MPC protocol uses the public value k to compare against the secret-computed threshold count:

$$alert = \begin{cases} 1 & \text{if } count \geq k \\ 0 & \text{otherwise} \end{cases}$$

The group alert reveals the least detailed output among the implemented functions, because it reveals only whether a collective warning condition has been met.

4.5 MPC Protocol Design in MP-SPDZ

As previously established, the collaborative assessment functions are implemented using MP-SPDZ. MP-SPDZ allows MPC programs to be written at a high level while the framework handles the underlying secure execution. Each organization is represented as a party, and its input values are provided through MP-SPDZ input files. The computations are then executed using the Shamir protocol.

As explained in the background section, the Shamir protocol is based on Shamir secret sharing. In the implementation, this means that the private input values are not processed as openly visible values. Instead, they are represented as secret-shared values inside MP-SPDZ. The assessment functions are then computed over these secret values, and only the final group-level outputs are revealed.

The aforementioned protocol was selected for the proof-of-concept because it compiled and executed successfully in the local WSL development environment, which is to be further explained in subsequent sections. An initial attempt to use the MASCOT protocol encountered dependency issues related to the local build environment. Since the goal of this thesis is to evaluate the feasibility of the assessment functions rather than compare all available MPC protocols, the Shamir protocol provides a suitable starting point for the proof-of-concept.

In the implementation, each party provides its private input values. These inputs are read as secret integers using MP-SPDZ. The assessment functions are computed over secret-shared values, and only the final group-level output is revealed using the `reveal()` operation. This means that intermediate values, such as individual weighted exposure scores, remain hidden during the computation.

4.6 Evaluation Metrics

The proof-of-concept is evaluated using four main metrics: correctness, runtime, communication cost, and scalability.

Correctness may be rather self-explanatory, as it aims to measure if the MPC output matches the expected plaintext result. For each implemented function, the same input values are computed both manually or using a plaintext baseline and through MP-SPDZ. The MPC result is considered correct if it matches the plaintext result.

Runtime measures how long the actual MPC execution takes. MP-SPDZ reports execution time after each run, including preprocessing. Using this metric, one is able to assess the computational overhead introduced by privacy-preserving computation.

Communication cost measures the amount of data exchanged between parties during the MPC execution. MP-SPDZ reports both the data sent by party 0 and the global data sent across all parties. This is important because MPC protocols often require communication between participants, and communication overhead can affect practical feasibility.

Scalability evaluates how performance changes as the number of participating organizations increases. In this thesis, scalability is tested by running the same MPC program with different values of n, where n represents the number of organizations participating in the collaborative assessment. Initial experiments are conducted with $n = 3$, $n = 5$, and $n = 10$. This allows the evaluation to observe how runtime, communication cost, and the number of communication rounds change as more parties participate in the protocol. Later experiments may attempt to push the limits of the system, as values increase to $n = 25$, $n = 50$, and as a final stress test $n = 100$. Using such values, this thesis is able to observe varying performance changes within functions. The expected comparison is that simple arithmetic functions, such as averages, require less overhead than comparison-based functions, such as threshold counts and group alerts.

Together, these metrics allow the thesis to evaluate whether privacy-preserving collaborative vulnerability exposure assessment can be implemented correctly and with acceptable computational and communication performance.

5 Implementation

This section describes the implementation of the proof-of-concept system. The purpose of the implementation is to translate the methodology described in Section 4 into an executable MPC experiment. The implementation consists of a local development environment, a synthetic data generator, MP-SPDZ input files, and a variable-n MPC program that computes collaborative vulnerability exposure metrics.

5.1 Development Environment

The proof-of-concept was implemented locally using MP-SPDZ in a Linux environment through Windows Subsystem for Linux (WSL). The MP-SPDZ repository was cloned into the WSL file system and opened in Visual Studio Code. Commands were executed through the integrated WSL terminal. All MPC parties were run locally on the same machine. This was done because the goal of the implementation was to evaluate the correctness and scalability behaviour of the selected assessment functions in a controlled proof-of-concept setting, rather than to deploy a complete real-world multi-organization system. Running the parties locally also made it possible to repeat experiments with different numbers of simulated organizations while keeping the software environment, input generation, and execution procedure consistent.

This local setup still simulates the multiparty setting because MP-SPDZ starts a separate party process for each participating organization. Each party receives its own private input file and participates in the protocol execution. However, because all party processes run on the same machine, the measurements should be interpreted as local experimental benchmarks rather than as direct predictions of performance in a distributed deployment across different organizations and networks.

An initial attempt was made to run the implementation using MP-SPDZ's MASCOT protocol. As explained in the background section, MASCOT is a malicious-secure arithmetic MPC protocol based on oblivious transfer [11]. However, this did not function properly in the local environment, as the local build encountered dependency-related compilation issues. Hence, the Shamir protocol was substituted instead. As also discussed in the background section, Shamir-based protocols rely on secret sharing, where private values are represented through shares rather than being directly revealed [15]. The Shamir protocol was able to successfully compile and execute in the local development environment, making it suitable for the proof-of-concept implementation. Since the main aim of the thesis is to evaluate whether collaborative vulnerability exposure functions can be implemented and benchmarked in MPC, the use of Shamir is sufficient for the experimental prototype. A comparison between different MPC protocols is left outside the scope of this thesis.

Before running experiments with a specific number of parties, SSL files were generated using the MP-SPDZ setup script. SSL files allow for the synthetically generated participants to locally establish authenticated communication channels during execution of the protocol. For example, experiments with n organizations required the following setup step:

```
Scripts/setup-ssl.sh n
```

After this, the MPC programs were compiled and executed using MP-SPDZ's `compile-run.py` script.

5.2 Synthetic Data Generator

Real organizational vulnerability and risk data is incredibly difficult to obtain due to its sensitive nature, as leakages may lead to the vast amount of detriments described previously. Hence, synthetic data was used for the experiments. A Python script was written to generate artificial company-level vulnerability exposure data. The number of organizations is controlled using a parameter n, where each organization corresponds to one MPC party.

For each organization, the script generates the four aforementioned integer input values:

- `cvss_score`: a CVSS-like vulnerability severity score scaled from 0 to 100;
- `critical_vulnerabilities`: the number of unpatched critical vulnerabilities;
- `patch_delay`: the average patch delay in days;
- `asset_exposure`: an exposure score representing public-facing or business-critical assets, scaled from 0 to 100.

The dataset generator is able to output various exposure scenarios, such as low, mixed, or high exposure. In the context of this thesis, the mixed scenario is used as the main setting. This is due to the fact that such scenario generates a combination of lower and higher exposure values, allowing for testing threshold-based functions and group alerts.

The script also computes plaintext expected results. This includes factors such as the weighted exposure score, average score, maximum score, threshold count, and group alert. These plaintext results are mainly utilized to verify the correctness of the actual MPC outputs.

5.3 MP-SPDZ Input Format

MP-SPDZ reads private inputs from files stored in the `Player-Data` directory. Within this directory, each organization is represented as one party, and each party receives its own input file. As demonstration, party 0 receives input through:

```
Player-Data/Input-P0-0
```

Similarly, party 1 receives input through:

```
Player-Data/Input-P1-0
```

The aforementioned integer values are represented in the following order:

```
cvss_score critical_vulnerabilities patch_delay asset_exposure
```

For example, an input file may contain:

```
80 5 30 70
```

This represents an organization with a scaled CVSS-like score of 80, five critical vulnerabilities, an average patch delay of 30 days, and an asset exposure score of 70. Such values are deemed as classified. Hence, the MPC program processes them as secret inputs, meaning that they are not revealed to the other parties during the computation.

5.4 Variable-n MPC Program

In order to observe functionality and usability, the first proof-of-concept programs were written for solely three parties. These initial programs computed simple functions such as an average exposure score, a weighted average exposure score, a threshold count, and a group alert. After these initial tests succeeded, the implementation was extended to support a variable number of organizations.

Hence, the final MPC program accepts three arguments:

- n: the number of organizations participating in the computation;
- T: the high-exposure threshold;
- k: the minimum number of high-exposure organizations required to trigger a group alert.

For each organization i, the MPC program reads four private inputs and computes the weighted exposure score:

$$score_i = 4 \cdot cvss_i + 2 \cdot critical_i + 2 \cdot patch_i + 2 \cdot exposure_i$$

The program then computes the following group-level outputs:

- the average weighted exposure score;
- the number of organizations with $score_i \geq T$;
- a group alert indicating whether at least k organizations exceed the threshold;
- the maximum weighted exposure score.

Following this, the MPC reveals only those final outputs. Naturally, individual input values and intermediate weighted scores remain hidden. This reflects the intended privacy-preserving setting: participating organizations learn collective vulnerability exposure information without seeing each other's private data.

5.5 Experimental Procedure

The experimental procedure consists of four major steps. First, SSL files are generated depending on the amount of participating parties. These files are required by MP-SPDZ to set up authenticated communication between the separate party processes. In the local experiment, each simulated organization is represented by a separate party process, and the SSL files allow these parties to identify and communicate with one another during the execution of the protocol. The SSL files do not contain the private vulnerability input values themselves, but are part of the communication setup required before the MPC computation can be executed. Second, the synthetic data generator creates the input files for n organizations. Third, the variable-n MPC program is compiled and executed using MP-SPDZ. Fourth, the output of the MPC program is compared with the plaintext expected results generated by the Python script.

A typical experiment with five organizations is executed within the WSL environment, and goes as follows:

```
Scripts/setup-ssl.sh 5
python3 thesis_scripts/generate_synthetic_inputs.py --n 5 --scenario mixed --threshold 500
Scripts/compile-run.py shamir thesis_variable_n 5 500 2 -- -N 5
```

The same procedure is applied to all varying values of n. In this thesis, the successful scalability experiments use $n = 3$, $n = 5$, $n = 10$, $n = 25$, and $n = 50$. A larger stress test with $n = 100$ was also attempted, but this exceeded the practical limitations of the local WSL environment. This stress test is discussed separately in the results and discussion sections.

For the experiments which were successful, the following information is output is recorded and will be analyzed later:

- the number of organizations;
- the average weighted exposure score;
- the number of organizations above the threshold;
- the group alert output;
- the maximum weighted exposure score;
- runtime;
- data sent by party 0;
- global data sent across all parties;
- approximate number of communication rounds.

These measurements form the basis for the experimental results in Section 6.

6 Experiments and Results

This section ultimately presents the experimental results of the MPC proof-of-concept. As pre-described, the main goal of the experiments was to evaluate whether the proposed collaborative assessment functions could be computed correctly and to observe how runtime and communication costs changed as the number of participating organizations increased.

The code used for the MPC implementation, synthetic data generation, and experimental runs is available in a public Git repository.[1] This repository contains the MP-SPDZ source program, the Python script used to generate synthetic inputs, and the recorded experimental output files used for the results reported in this section. This repository contains the MP-SPDZ source program, the Python script used to generate synthetic inputs, and the recorded experimental output files used for the results reported in this section.

The experiments were carried out in two stages. Initially, a small proof-of-concept was tested with simply three organizations. This was mainly done to verify the basic correctness of the implementation.

[1] https://github.com/Donplaynl/thesis-mpc-vulnerability-assessment

Subsequently, the implementation was extended to a variable-n setting and tested with $n = 3$, 5, 10, 25, and 50 organizations. The values 3, 5, and 10 may be considered more realistic for smaller collaborative settings, while 25 and 50 were included to explore the scalability limits of the implementation. Finally, a stress test with $n = 100$ organizations was attempted.

6.1 Initial Three-Party Proof of Concept

The first successful experiments were performed with three organizations. The main purpose of such initial tests was to confirm that the private input pipeline, weighted score computation, threshold-based functions, and final reveal operations all worked correctly. In context of the thesis, this meant that the MPC program could compute group-level outputs without revealing individual organizational inputs.

The initial three-party experiments provided confidence that the proof-of-concept implementation was correct and motivated the extension to the variable-n setting.

6.2 Variable-n Scalability Experiment

The main experiment used the variable-n MPC implementation described in Section 5. For each value of n, the program computed the following group-level outputs:

- the average weighted exposure score;
- the number of organizations above the threshold $T = 500$;
- the group alert output;
- the maximum weighted exposure score.

The minimum number of high-exposure organizations required to trigger a group alert was scaled with the experiment size. Specifically, the values $k = 2, 2, 3, 8$, and 15 were used for $n = 3, 5, 10, 25$, and 50, respectively.

Table 1 summarizes the results of the successful runs.

n	Avg. score	Above	Alert	Max	Time (s)	Data (MB)	Global MB
3	480.00	1	0	644	0.0108	0.0488	0.1273
5	438.40	1	0	644	0.0286	0.1902	0.8468
10	449.80	3	1	644	0.1008	0.8187	4.5272
25	474.72	9	1	644	1.1657	4.5909	62.2170
50	495.28	25	1	684	5.9474	18.1396	474.6280

Table 1: Scalability results for the variable-n MPC implementation using the Shamir protocol.

The table shows that the implementation produced meaningful outputs for all successful experiment sizes. The average weighted exposure scores remained within a comparable range across runs, while the threshold count and group alert changed depending on the generated synthetic inputs and the chosen alert parameter k. More importantly, the runtime and communication measurements increased substantially as the number of organizations grew.

6.3 Runtime Results

Figure 1 shows the runtime of the MPC implementation as the number of organizations increases.

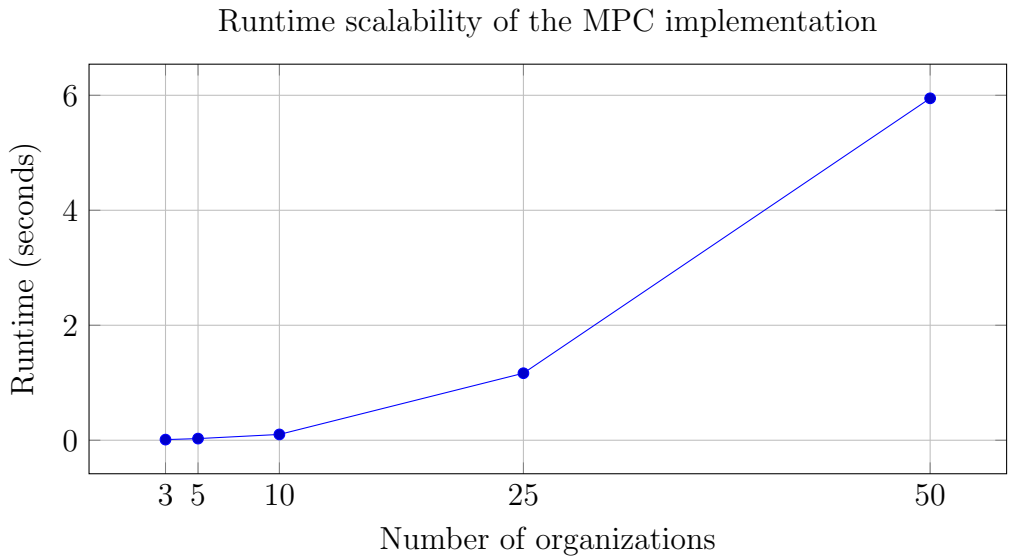

Figure 1: Runtime of the variable-n MPC implementation using the Shamir protocol.

Initially, the runtime increases rather slowly. However, as the number of organizations grew, the runtime grew substantially larger. As can be observed, the runtime for $n = 50$ is around 550 times longer than for $n = 3$. These runtime values should be interpreted as local experimental benchmark results, since all parties were executed as separate processes on the same machine. No additional network delay or bandwidth limitation was simulated. Therefore, the results show how the implementation behaves in a controlled local setting, but they should not be interpreted as direct predictions of runtime in a real distributed environment. In such a deployment, communication latency between organizations would likely increase the total runtime further.

6.4 Communication Cost Results

Figure 2 shows the total global communication cost for each successful run.

As can be observed, the communication overhead grows very rapidly as the number of organizations increases. This means that adding more parties does not only add a small amount of extra communication, but instead leads to a much larger total amount of data being exchanged between parties. This is especially visible when moving from $n = 10$ to $n = 25$, and again from $n = 25$ to $n = 50$. Compared to runtime, communication appears to be an even more severe bottleneck. These results indicate that while the proof-of-concept is feasible, communication cost becomes a major limitation when many organizations participate.

6.5 Communication Rounds

In addition to runtime and communication volume, the approximate number of communication rounds also increased with the number of organizations. Figure 3 shows this trend.

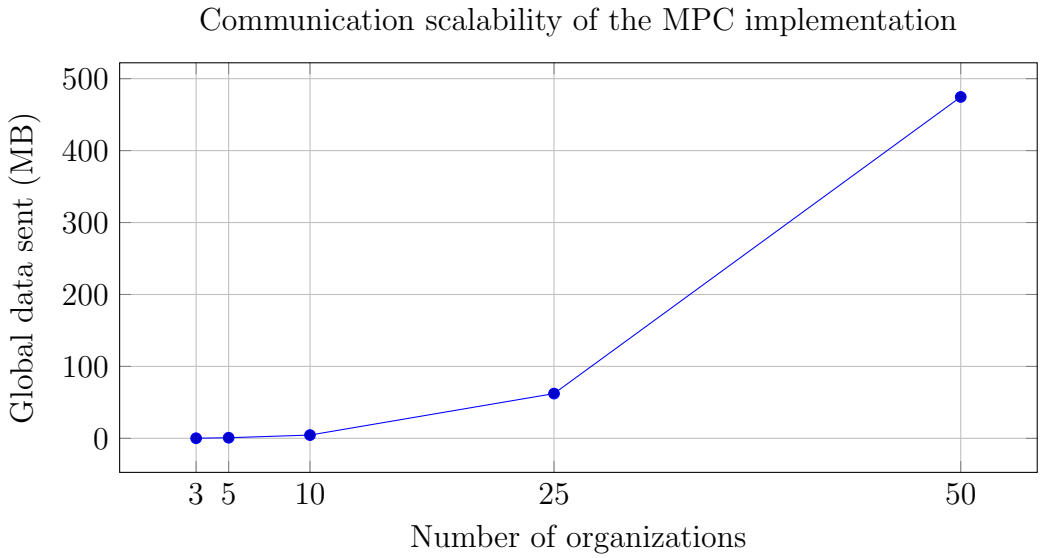

Figure 2: Global communication cost of the variable-n MPC implementation using the Shamir protocol.

The increasing number of rounds provides further evidence that the protocol becomes less efficient as more parties participate. This is consistent with the runtime and communication measurements shown above.

6.6 Stress Test with 100 Organizations

After the initial values were computed successfully, a final stress test was attempted with $n = 100$ organizations. A number of processes were successful, such as the compilation, and the synthetic input files were generated correctly. However, execution did not complete successfully in the local WSL environment. An output error message occurred, which indicated that the system could not start the required number of threads for all participating parties.

This result suggests that the limitation at $n = 100$ was not necessarily caused by the computation itself, but by the practical resource limits of running a large number of local MPC party processes in the development environment. Nevertheless, the failed 100-party run is an important result, because it demonstrates that the current Shamir-based proof-of-concept and local setup do not scale indefinitely.

Overall, the largest successful experiment in this thesis used $n = 50$ organizations. The results show that the implementation works correctly and provides meaningful group-level outputs, but that both runtime and communication cost increase substantially as the number of participating organizations grows.

7 Discussion

This section primarily discusses the results of the MPC proof-of-concept and relates them back to the research question. As established, the aim of the thesis was to investigate whether collaborative

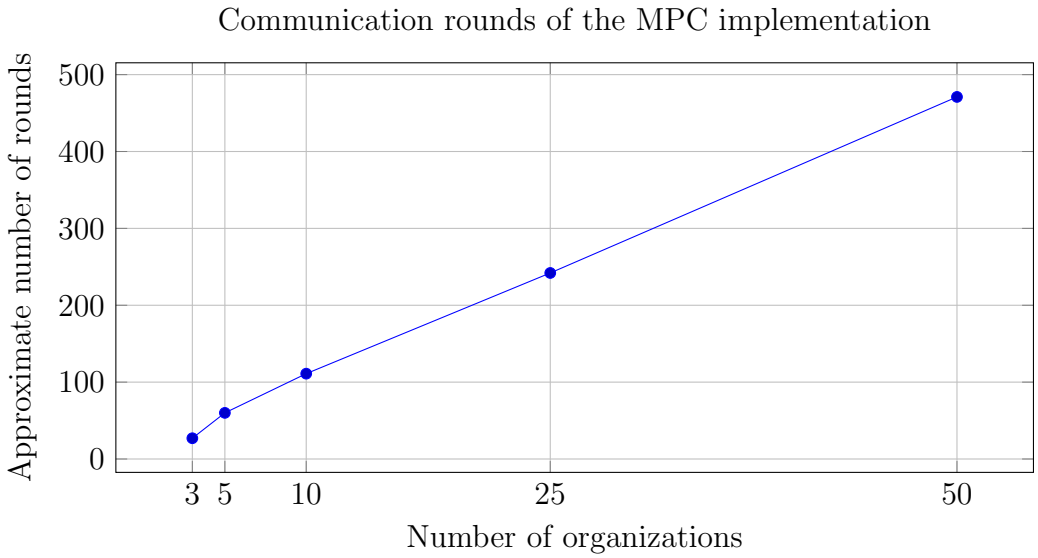

Figure 3: Approximate number of communication rounds for the variable-n MPC implementation.

vulnerability or risk assessment across multiple organizations can be designed and implemented using Secure Multi-Party Computation while preserving private organizational inputs and maintaining acceptable computational and communication performance. The results show that such a proof-of-concept is feasible, but also that performance costs become increasingly important as the number of participating organizations grows.

7.1 Interpretation of Results

The experiments demonstrate that the implemented MPC program can successfully compute group-level vulnerability exposure metrics over private organizational inputs. For each organization, synthetic vulnerability indicators were used to compute a weighted exposure score. The MPC program then computed the average weighted exposure score, the number of organizations above a predefined threshold, a group alert, and the maximum weighted exposure score.

The most important result is that these outputs were computed without revealing the individual input values of the participating organizations. This matches the main motivation of the thesis: organizations can obtain useful collaborative security information without directly sharing sensitive vulnerability or risk data. For example, the threshold count reveals how many organizations exceed a high-exposure threshold, but does not reveal which organizations those are. Similarly, the group alert only reveals whether the number of high-exposure organizations is large enough to trigger a collective warning.

The three-party proof-of-concept verified the basic pipeline of the implementation. After this, the variable-n experiments showed that the same implementation could be extended to larger numbers of organizations. Successful runs were completed for $n = 3$, $n = 5$, $n = 10$, $n = 25$, and $n = 50$. This suggests that the implemented functions are not limited to a fixed number of participants and that the design can be scaled at least within the limits of the local experimental environment.

7.2 Privacy and Performance Trade-offs

The results also show a clear trade-off between privacy and performance. In a non-private setting, the weighted exposure scores and group metrics could be computed very cheaply by collecting all input data in one central location. However, this would require organizations to reveal sensitive information about their vulnerability exposure, patching delays, and asset exposure. MPC avoids this direct disclosure, but introduces additional computational and communication overhead.

Runtime increased as the number of organizations grew. For small settings, such as $n = 3$, $n = 5$, and $n = 10$, the runtime remained low. This suggests that MPC can be practical for small collaborative settings, such as a small group of organizations in the same sector or supply chain. However, runtime increased more strongly for $n = 25$ and $n = 50$. This indicates that larger collaborative settings require more careful protocol selection and optimization.

As observed, the cost attached to communication increase even more strongly than runtime. Initially, the global data sent was less than one megabyte, which then grew to hundreds of megabytes in subsequent experiments. This indicates that communication overhead is one of the main practical limitations of the implemented approach. Especially in real-world scenarios, communication costs may prove increasingly relevant, as organizations would not necessarily run on the same machine or local network. Network latency, bandwidth limitations, and coordination between organizations could all increase the practical cost of running such protocols.

The results therefore support a nuanced answer to the research question. The implementation shows that privacy-preserving collaborative vulnerability exposure assessment is possible and works correctly for the tested functions. However, acceptable performance depends strongly on the number of participants, the chosen MPC protocol, and the complexity of the computed function.

7.3 Limitations

Several limitations should be considered when interpreting the results. First, the input data used in the experiments is synthetic. Due to the nature of real-world data, this was a deliberate design choice. Hence, although synthetic data makes it possible to test the MPC functions in a controlled way, it does not fully represent the complexity of real cybersecurity environments.

Second, the weighted exposure score used in the implementation is rather simplified. When looking at such a score, it combines CVSS-like score, number of critical vulnerabilities, patch delay, and asset exposure using fixed weights. For the use of experimentation it proves useful, but it should not be interpreted as a validated or complete cyber risk model. For example, additional information may be required for risk assessment, such as exploit availability, asset criticality, compensating controls, business impact, sector-specific requirements, etc.

Third, the implementation assumes a semi-honest threat model. Within this model, it suggests that parties are assumed to follow the protocol correctly, despite trying to possibly learn or gain information from outputs or messages. Under these assumptions, this thesis does not consider malicious parties which may submit false inputs, deviate from the protocol, or attempt to disrupt computation. Hence, this limits the strength of the security assumptions in the current proof-of-concept.

Fourth, the experiments were carried out in a local WSL environment. Although this allowed controlled testing, it should be noted it differs from real-world deployment, in which organizations would run on separate machines and communicate over a network. Consequently, the currently

measured performance should rather be interpreted as an experimental benchmark, as opposed to a direct prediction of deployment performance.

Finally, in the end only one major MPC protocol was used for experimentation. This is mainly due to MASCOT encountering issues regarding dependencies. Regardless, this limits the ability to generalize the performance results to all MPC protocols. Different protocols may perform better or worse depending on the number of parties, security assumptions, network setting, and type of computation.

7.4 Implications for Collaborative Security Assessment

Despite such limitations, the results show that MPC is a promising technique for privacy-preserving collaborative security assessment. When carrying out the experiments, the proof-of-concept demonstrates that organizations can jointly compute useful group-level metrics without revealing their individual vulnerability indicators. This is especially relevant in settings where organizations have a shared interest in understanding collective exposure, but cannot directly disclose internal security data.

In this context, the threshold count and group alert functions are particularly meaningful. This can be deduced, as they allow organizations to obtain the necessary information to determine risk conditions existing. Such functions could support sector-level security coordination, supply-chain risk monitoring, or collaborative vulnerability prioritization.

At the same time, the results show that MPC should not be treated as a cost-free solution. The communication overhead grows rather quickly as the number of organizations increases, especially under the Shamir protocol used in this thesis. Therefore, practical deployments would need to carefully choose the computed functions, the number of participants, the MPC protocol, and the deployment environment.

Overall, the findings suggest that MPC can support privacy-preserving collaborative vulnerability and risk assessment in principle. The implemented proof-of-concept provides evidence that selected assessment functions can be computed correctly and privately, while the scalability results highlight the performance challenges that must be addressed before such systems can be used at larger scale.

8 Conclusions and Further Research

8.1 Answer to the Research Question

To provide an answer to the main research question at hand, some aforementioned topics and points of discussions are notable. To initialize these discussion points, this thesis aimed to investigate how collaborative vulnerability or risk assessment across multiple organizations can be designed and implemented using Secure Multi-Party Computation while preserving each participant's private security data and maintaining acceptable computational and communication performance.

As initial discussion point, the proof-of-concept shows that this is possible in a simplified experimental setting. Through the use of MP-SPDZ and mainly the Shamir protocol, multiple metrics were able to be jointly computed over private input data. The major implemented functions include an average weighted exposure score, a threshold count, a group alert, and a maximum weighted exposure score. Due to the nature of the experiment, only the outputs of all functions were revealed.

The results garnered a number of interesting discussion points. Firstly, they showed that the approach is fully feasible within a small collaborative setting. In this case, numbers such as $n = 3$, $n = 5$, and $n = 10$ revealed both low runtime and communication overload, whilst still executing successfully. On the other hand, larger values such as $n = 25$ and $n = 50$ showed substantial increase in runtime and especially communication costs, despite still running successfully. Therefore, the answer to the research question is that MPC can support privacy-preserving collaborative vulnerability assessment, but its practical feasibility depends strongly on the number of participants, the chosen protocol, and the communication overhead.

8.2 Main Contributions

This thesis makes three main contributions. As a prerequisite to experimentation, it connects privacy-preserving computation techniques to the specific setting of collaborative vulnerability and risk assessment. Hence, Instead of focusing only on threat indicator sharing, the thesis considers numerical assessment functions over vulnerability-related security indicators.

Second, it defines and implements a proof-of-concept system in MP-SPDZ. Utilizing synthetic datasets for organizational inputs, it computes group-level metrics. The aforementioned functions demonstrate how private organizational security indicators can be transformed into collaborative outputs, without unecessary direct data sharing.

Third, the thesis evaluates the proof-of-concept experimentally. The results are able to be interpreted in various ways previously discussed. Ultimately, it shows that the implementation works correctly for multiple numbers of organizations and provide insight into runtime, communication cost, and scalability. These results show both the potential and the practical limitations of using MPC for collaborative security assessment.

8.3 Future Research

Future research could extend this work in several directions. As discussed, this thesis only utilizes the Shamir protocol. Hence, alternative MP-SPDZ protocols could be evaluated and compared. For example, ATLAS is an MPC protocol designed for efficient and scalable computation in honest-majority settings [7], while MASCOT is a malicious-secure arithmetic MPC protocol based on oblivious transfer [11]. Comparing such protocols would make it possible to determine whether different security assumptions or protocol designs scale better for larger numbers of organizations.

Second, the synthetic risk model could be improved by using more realistic vulnerability assessment data from real-world companies, or more domain-specific scoring methods. The weighted score used in this thesis is useful for benchmarking, but it is not intended as a complete real-world cyber risk model.

Third, future work could consider stronger security assumptions. This thesis assumes a semi-honest threat model, but real-world collaborations may include participants that submit incorrect inputs or deviate from the protocol. Studying malicious-security MPC protocols would therefore be an important extension.

Finally, the experiments were carried out on one system and through one terminal. Hence, the implementation could be tested in a distributed environment across multiple machines. This would better reflect a realistic setting in which independent organizations participate from separate networks and would provide more accurate insight into real-world communication costs.

References

[1] Hadeel Al-Issa, Mohammad A. Ottom, and Azzam Tamrawi. ehealth cloud security challenges: A survey. *Journal of Healthcare Engineering*, 2019.

[2] Onur Eren Arpaci, Raouf Boutaba, and Florian Kerschbaum. Over-threshold multiparty private set intersection for collaborative network intrusion detection. In *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*, pages 2211–2226, Renton, WA, May 2026. USENIX Association.

[3] Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. In *Computer Security – ESORICS 2008*, volume 5283 of *Lecture Notes in Computer Science*, pages 192–206. Springer, 2008.

[4] Cybersecurity and Infrastructure Security Agency. Known exploited vulnerabilities catalog, 2026. Accessed: 2026-05-31.

[5] Joint Task Force. Risk management framework for information systems and organizations: A system life cycle approach for security and privacy. Technical Report NIST Special Publication 800-37 Revision 2, National Institute of Standards and Technology, 2018.

[6] Forum of Incident Response and Security Teams. Common vulnerability scoring system sig, 2024. Accessed: 2026-05-31.

[7] Vipul Goyal, Hanjun Li, Rafail Ostrovsky, Antigoni Polychroniadou, and Yifan Song. Atlas: Efficient and scalable mpc in the honest majority setting. In *Advances in Cryptology – CRYPTO 2021*, volume 12826 of *Lecture Notes in Computer Science*, pages 244–274. Springer, 2021.

[8] Joint Task Force Transformation Initiative. Guide for conducting risk assessments. Technical Report NIST Special Publication 800-30 Revision 1, National Institute of Standards and Technology, 2012.

[9] Christopher Johnson, Lee Badger, David Waltermire, Julie Snyder, and Clem Skorupka. Guide to cyber threat information sharing. Technical Report NIST Special Publication 800-150, National Institute of Standards and Technology, 2016.

[10] Marcel Keller. Mp-spdz: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1575–1590, 2020.

[11] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Mascot: Faster malicious arithmetic secure computation with oblivious transfer. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 830–842, 2016.

[12] Yehuda Lindell. *Secure Multiparty Computation*. Morgan & Claypool, 2020.

[13] Rasoul Akhavan Mahdavi, Thomas Humphries, Bailey Kacsmar, Simeon Krastnikov, Nils Lukas, John A. Premkumar, Masoumeh Shafieinejad, Simon Oya, Florian Kerschbaum, and Erik-Oliver Blass. Practical over-threshold multi-party private set intersection. In *Proceedings of the 36th Annual Computer Security Applications Conference*, pages 772–783, 2020.

[14] MISP Project. Sacti – secure aggregation of cyber threat intelligence, 2022. Accessed: 2026-05-31.

[15] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[16] Florian Skopik, Giuseppe Settanni, and Roman Fiedler. A problem shared is a problem halved: A survey on the dimensions of collective cyber defense through security information sharing. *Computers & Security*, 60:154–176, 2016.

[17] Murugiah Souppaya and Karen Scarfone. Guide to enterprise patch management planning: Preventive maintenance for technology. Technical Report NIST Special Publication 800-40 Revision 4, National Institute of Standards and Technology, 2022.

[18] Wiem Tounsi and Helmi Rais. A survey on technical threat intelligence in the age of sophisticated cyber attacks. *Computers & Security*, 72:212–233, 2018.

[19] Thijs Veugen, Gabriele Spini, and Frank Muller. Secure aggregation of sufficiently many private inputs. *Frontiers in Big Data*, 8, 2025.

[20] Andrew C. Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 160–164, 1982.