Security Games with Malicious Adversaries in The Clouds: Status Update

Artis Carter[a], Richard Hernandez[b], Soamar Homsi[c], and G. Makenzie Cosgrove[c]

[a]Augusta University, Augusta, GA, U.S.A
[b]Florida International University, Miami, FL, U.S.A
[c]Air Force Research Laboratory/ Information Directorate, Information Warfare Division, Rome, NY, U.S.A

ABSTRACT

Outsourcing computational tasks to the cloud offers numerous advantages, such as availability, scalability, and elasticity. These advantages are evident when outsourcing resource-demanding Machine Learning (ML) applications. However, cloud computing presents security challenges. For instance, allocating Virtual Machines (VMs) with varying security levels onto commonly shared servers creates cybersecurity and privacy risks. Researchers proposed several cryptographic methods to protect privacy, such as Multi-party Computation (MPC). Attackers unfortunately can still gain unauthorized access to users' data if they successfully compromise a specific number of the participating MPC nodes. Cloud Service Providers (CSPs) can mitigate the risk of such attacks by distributing the MPC protocol over VMs allocated to separate physical servers (i.e., hypervisors). On the other hand, underutilizing cloud servers increases operational and resource costs, and worsens the overhead of MPC protocols. In this ongoing work, we address the security, communication and computation overheads, and performance limitations of MPC. We model this multi-objective optimization problem using several approaches, including but not limited to, zero-sum and non-zero-sum games. For example, we investigate Nash Equilibrium (NE) allocation strategies that reduce potential security risks, while minimizing response time and performance overhead, and/or maximizing resource usage.

Keywords: Machine Learning, Optimization, Multi-party Computation, Secret Sharing, Game Theory

1. INTRODUCTION

Machine Learning (ML), a subset of artificial intelligence and computer science, utilizes data and algorithms to mimic human learning processes, aiming to progressively enhance its precision. ML enables computers to learn without explicit programming, using algorithms that learn from data. Outsourcing ML applications to the cloud has several substantial advantages. Firstly, it leverages the cloud's scalable and elastic infrastructure, allowing organizations to efficiently allocate computing resources based on the evolving needs of ML workloads.[1] This flexibility accelerates model training and deployment, reducing time-to-insight, which measures how quickly data is turned into actionable insights. Cloud-based ML services often incorporate powerful hardware and pre-configured environments, simplifying the development process and reducing infrastructure-related complexities. However, entrusting ML applications to cloud environments also introduces inherent security challenges.[2,3] Relinquishing control over critical components, including data storage and processing, introduces security concerns. For example, data privacy becomes a paramount issue when sensitive information is involved, as sharing data with third-party Cloud Service Providers (CSPs) raises questions about confidentiality. These concerns prompt organizations to balance the benefits of cloud outsourcing with stringent privacy measures while maintaining data confidentiality and trust. Cryptographic techniques and protocols, such as Multi-party Computation (MPC), have emerged as powerful tools to address the security challenges cloud computing poses.[4] MPC allows multiple

Further author information: (Send correspondence to S.H.)
A.C.: E-mail: artcarter@augusta.edu
R.H.: E-mail: rhern336@fiu.edu
S.H.: E-mail: soamar.homsi@us.af.mil
M.C.: E-mail: gage.cosgrove@us.af.mil

Approved for Public Release on 08 Mar 2024. Distribution Is Unlimited. Case Number: AFRL-2024-1317.

parties to jointly compute a function over their inputs while keeping them private.[5] In ML, various parties can collaborate to train a machine learning model on their combined data without any party needing to reveal their data to others. MPC addresses the privacy concern, as data is kept confidential, and only the output is revealed. Moreover, MPC also provides robustness against specific adversarial attacks since the attacker would need to corrupt a majority of the parties to affect the computation via a simple denial of service (DoS) attack. However, MPC incurs computation and communication overhead, and it is susceptible to security breaches. For example, when attackers compromise enough number of MPC nodes, which we call the threshold , they can access the private data. Implementing ML training and inference across a wide network of MPC nodes can mitigate the risk of data compromise. Unfortunately, this approach significantly increases resource consumption and results in lower utilization rates of cloud resources. The MPC deployment in the cloud necessitates greater bandwidth and computational power, leading to escalated costs and potentially unsustainable resource demands. In addition, the complexity of ML computations, characterized by intricate linear and nonlinear operations, such as matrix multiplications, increases the latency of MPC protocols. Especially for ML tasks, the execution times are substantially prolonged due to the computationally intensive nature of these operations across multiple nodes. The increased network latency can hinder real-time applications and diminish the overall efficiency of using MPC for ML in cloud environments. Acknowledging these challenges, recent research endeavors seek to refine MPC's application in ML, focusing on enhancing security protocols, optimizing resource utilization, and improving response times, thus paving the way for more efficient and secure computing solutions.

Recent studies focus on overcoming key obstacles that limit the application of MPC in ML, including stringent security requirements (i.e., threat model) as outlined by Lindell,[5] intensive use of computational resources (CPU, RAM, storage, and network), and prolonged protocol execution times. To address the security challenges, several works[6–8] offer better overall performances by reducing the security requirements of the ML training. These security requirements are commonly described by the number of malicious participants the protocol can protect against, i.e. the threshold number of malicious MPC nodes. Current studies[9–11] focus on specific operations from ML and reduce their performance footprint using more efficient algorithms to achieve reductions in the computation and communication resources. For example, "matrix triples" introduced by Chen et al.[9] use additive homomorphic encryption schemes to improve the network overhead and execution time of matrix multiplication, a crucial part of machine learning. Others[10] utilize the Strassen algorithm to reduce communication overhead and execution time. These operational optimizations typically introduce more parameters, which vary based on the application, so most of these works require parameter optimization to help reduce these challenges. Unfortunately, selecting such parameters is challenging. Tools such as game theory can be employed to determine the optimal parameters for different applications efficiently. To address MPC limitations in ML applications, we adopt game theory that emerges as a compelling analytical tool and strategic decision-making approach theory to navigate the landscape of cybersecurity challenges, inefficient resource allocation, and lengthy response time.

Game theory is an area of applied mathematics that focuses on the analysis of actions and strategies for dealing with competitive situations where the outcomes for each participant depend on the actions of all. We employ game theory to model the cybersecurity loss problem, as it provides a structured framework for analyzing strategic interactions between intelligent and rational decision-makers.[12] By applying game-theoretic principles, we can gain insights into optimal decision-making processes, which makes it a valuable tool for assessing and mitigating the risks of outsourcing ML applications to the cloud. Homsi et. al[13,14] used game theory to minimize a CSP's potential security loss when allocating several virtual machines on a cloud cluster. This approach is particularly well-suited for understanding the dynamics between a CSP and an attacker because it allows for the exploration of various strategies and their potential outcomes. Specifically, they investigated the optimization of VM allocation in cloud clusters to balance cybersecurity risks and operational costs, including energy consumption, and here considered the case where there are more than two servers. Their approach to this multi-objective optimization problem involves modeling the conflict between CSPs and attackers as a non-cooperative, zero-sum game to minimize the service provider's security loss, before then incorporating factors like resource utilization, power consumption, and costs into a non-zero-sum game model. They aimed to find the optimal Nash equilibrium strategies, but not in the context of an MPC scenario. We also seek to develop a novel heuristic for finding optimal equilibrium strategies quickly and more efficiently when solving with classical algorithms proves infeasible or inefficient, but under the malicious security assumptions in an MPC setting.

In this paper, we describe our ongoing research that investigates secure static VM allocation problems for

executing MPC in commercial clouds and present the following contributions. In Problem I, we assume a resource-unconstrained CSP where the sole focus is to minimize security loss due to a data breach during an MPC protocol; we model the problem as a zero-sum game and introduce a heuristic that can enumerate and produce all optimal allocation strategies quickly and more efficiently than using brute force or classical Game Theory algorithms. In Problem II, we consider a resource-constrained CSP who wants to achieve the same goal of Problem I. After formulating this problem as a zero-sum game, we enumerate the total number of possible allocations and calculate the potential minimum and maximum loss under each of the scenarios when servers and VMs are identical or different. However, the work to find a solution is ongoing.

For the third and final problem, we consider other CSP operational factors, such as network bandwidth and computational cost. Specifically, we model this problem as a non-zero-sum game and use game theory to minimize execution time and resource cost simultaneously. We will then compare this solution to a brute force and Non-dominated Sorting Genetic Algorithm (NSGA-II)[15] methods.

2. BACKGROUND

Multi-objective optimization (MOO) entails maximizing or minimizing multiple objectives subject to constraints. MOO problems can be solved in various ways, such as using genetic algorithms, machine learning, and mathematical programming. Mathematical programming involves the use of mathematical models to solve problems. The mathematical programming techniques used to solve MOO problems transform the problem into a single-objective problem that can be solved using traditional optimization methods, such as the weighted sum approach and the ϵ-Constraint method.[13] In contrast, genetic algorithms evaluate each individual's fitness in a population, culling the population by retaining the elite (i.e., individuals with the highest fitness), which are then used to create a new population.[16] Machine learning can use training methods such as reinforcement and deep learning to solve MOO problems.[17] For example, multi-objective reinforcement learning (MORL) prompts agents to simultaneously optimize multiple objectives by learning policies (i.e., strategies). Multi-task Learning, a subfield of deep learning, is a method of training one model to perform multiple tasks (i.e., optimizing multiple objectives). While these methods are useful for solving MOO problems, they do not provide a way to deal with the behaviors of cybersecurity attackers. By contrast, game theory allows us to optimize while considering the response from an attacker, specifically, by viewing security as one of the objectives to be minimized.

We choose to model our research problem as a two-player non-cooperative, zero-sum, and non-zero-sum game models with perfect information. That is, there are two players and both have an objective that is contrary to the objective of the other player (i.e., non-cooperative). Furthermore, any action resulting in a gain for a given player results in an identical loss for the other player (i.e., zero-sum). Finally, each player is aware of all possible actions the other player can take, as well as any actions they take. We assume that each player is rational (i.e. that they will take the best possible action for themselves given their respective objectives). By analyzing the interactions and strategies of rational decision-makers, such as a service provider and attacker, game theory identifies strategies that a provider can employ to minimize security losses from successful attacks. Building on the premise of rationality and strategic interaction in game theory, we now turn to the concept of equilibrium as established by Nash. In our case, the NE is the strategy that conforms to an optimal VM allocation strategy which minimizes security loss for a CSP.

2.1 Nash Equilibria and Payoff Matrices

Nash proved in his thesis that every finite, n-player, non-cooperative game, including both zero-sum and non-zero-sum games, has at least one equilibrium point. This equilibrium, known as Nash equilibrium (NE), consists of strategies where no player benefits from unilaterally changing their strategy, providing a foundational framework for understanding strategic decision-making in competitive and cooperative scenarios.[12] In our case, we choose to model the worst-case scenario cybersecurity loss of any action taken. This is critical because it is unfair to consider optimal any strategy that can be exploited by a clever attacker. Specifically, a CSP is said to be using an optimal strategy if the CSP receives no additional benefit from choosing another strategy. However, we assume that the attacker is rational and will always act to maximize their gain. As such, if the CSP is acting optimally, then the attacker is as well and will receive no additional gain from altering their strategy. In this case, we have described precisely a NE. It is possible that, for our game, there are no NEs or that there is more than one NE

and, moreover, there is no guarantee that a NE is optimal. However, when an NE exists, the above shows that the optimal CSP action is among the NE strategies, so it suffices to determine when NE strategies exist, identify NE strategies, and then determine which is optimal.

The game-theoretic model creates a matrix whose rows represent strategies taken by the CSP and whose columns represent strategies taken by an attacker. The entries of the matrix represent the gain the attacker receives by employing a particular attack strategy when the CSP employs a particular strategy. This matrix is referred to as the payoff matrix for the game. It is not clear at present that this matrix is finite, however, when it is, we may locate a Nash equilibrium for the underlying game by finding the maximum entry in each row (this models the attacker choosing their best strategy for each CSP action) and then finding the row with the minimal such entry (this models the CSP choosing the optimal strategy knowing the attacker's best reply). Note that the NE may not be optimal for any player because it represents a situation where no player can benefit by changing their strategy alone. It does not guarantee the best possible outcome for either player individually. This dichotomy arises because the NE focuses on stable outcomes rather than individual player optimization. In the following section, we present our initial problem aimed at minimizing a cloud service provider's security loss in the event of a successful attack by an adversary, while not considering resource utilization and response time.

3. PROBLEM I

3.1 System Model

Users outsource their data to the cloud to perform secure computation using cloud resources (e.g., VMs) and cloud services (e.g., MPC). CSPs offer users compensation in the case of a data breach. We assume that a CSP manages and runs a cluster of N cloud servers. Each server contains a hypervisor, H_i, that can host any number of VMs. The probability that an attacker can successfully compromise the hypervisor H_i is q_i, where $0 < q_i < 1$. We assume that adversaries must directly attack a hypervisor before they can compromise the VMs hosted on that hypervisor. However, once a hypervisor is successfully attacked, any number of VMs being hosted on the hypervisor may be attacked, i.e. the hypervisor need only be successfully attacked once.

The goal of the attacker is to gain unauthorized access to the secret-shared data which is the input data and information provided by the users. This requires the attacker to successfully compromise t VMs, where t is the threshold of the MPC protocol. When discussing the security of MPC, several properties are considered, such as fairness and guaranteed output delivery. Fairness means corrupted parties should receive their outputs if and only if the honest parties also receive their outputs.[5] Guaranteed output delivery means corrupted parties should not be able to prevent honest parties from receiving their output.[5] In other words, the adversary should not be able to carry out a denial of service attack to disrupt the computation. Here, we assume malicious security, meaning up to $t-1$ VMs can be malicious without breaking the security of the protocol. In other words, using MPC assuming malicious security settings ensures that the confidentiality of user's data is maintained so long as the number of compromised VMs does not exceed $t-1$.

In the literature, t is often considered to be within the following ranges:

- $1 \leq t \leq \frac{M}{3}$;
- $\frac{M}{3} \leq t \leq \frac{M}{2}$;
- $\frac{M}{2} \leq t \leq M$.

For $t < \frac{M}{3}$ (i.e., less than a third of MPC parties are corrupted), secure Multi-party protocols with fairness and guaranteed output delivery can be achieved for any function. This assumes computational security in a synchronous point-to-point network with authenticated channels,[18] and information-theoretic security with private channels.[19,20] Rabin's verifiable secret sharing (VSS) and MPC Protocol[21] and the CCD (Chaum, Crépeau, Damgård) protocol[20] are some examples of MPC protocols with $t < \frac{M}{3}$.

Approved for Public Release on 08 Mar 2024. Distribution Is Unlimited. Case Number: AFRL-2024-1317.

When $t < \frac{M}{2}$ (i.e., in the case of a guaranteed honest majority), these protocols can be achieved with both computational and information-theoretic security, provided there is access to a broadcast channel.[18,22] The BGW (Ben-Or, Goldwasser, and Wigderson)[19] and GMW (Goldreich, Micali, and Wigderson)[18] protocols are examples of MPC protocols with such a threshold. However, for $t \geq \frac{M}{2}$ (i.e., when we have a malicious majority) secure Multi-party protocols are achievable, but without the assurance of fairness or guaranteed output delivery.[18,23] The Ishai, Prabhakaran, and Sahai (IPS) Protocol is one such protocol with $t \geq \frac{M}{2}$.[24]

We will index and refer to VMs as VM_j. We denote the probability that an attacker can compromise VM_j by p_j, where $0 < p_j < 1$. A higher q_i or p_j corresponds to a lower security level, whereas a lower value indicates a higher security level.

Finally, the CSP can select a VM allocation strategy β, which is represented by an $N \times M$ matrix where the (i, j) entry is 1 if H_i hosts VM_j and 0 otherwise. Importantly, each VM is hosted by a single server, so that there is only a single non-zero entry in each column. We shall refer to the space of all possible allocation strategies as B. We denote by m_i the number of VMs managed by H_i. Consequently, we obtain

$$M = m_1 + m_2 + \cdots m_n.$$

In summary, the CSP is concerned with the following parameters:

- $N \geq 1$, the number of servers;
- $0 < q_i < 1$, the probability that H_i will be successfully attacked;
- $M \geq 2$, the number of VMs;
- $0 < p_j < 1$, the probability that VM_j will be compromised by an attacker;
- $t \geq 1$, the threshold based on the MPC protocol;
- β, the allocation strategy for placing VMs on servers; and
- C, the financial compensation owed if the data is compromised.

3.1.1 The Gain from an Attack

Once the CSP sets their parameters, the attacker then carries out their attack. We model an attack strategy α as an $N \times M$ matrix whose (i, j) entry is 1 if the attacker targets VM_j being hosted by H_i and 0 otherwise. Throughout this paper, we shall denote by A the space of all possible attack strategies. We set

$$P = \begin{bmatrix} p_1 & p_2 & \cdots & p_M \\ \vdots & \vdots & \ddots & \vdots \\ p_1 & p_2 & \cdots & p_M \end{bmatrix}$$

to be the $N \times M$ matrix of probabilities of compromising a VM and define

$$\widehat{Q} = \begin{bmatrix} \widehat{q_1} \\ \widehat{q_2} \\ \vdots \\ \widehat{q_N} \end{bmatrix}$$

to be the $N \times 1$ vector of probabilities of successfully attacking a server with

$$\widehat{q_i} = \begin{cases} 0, & \text{if the } i^{\text{th}} \text{ row of } \alpha \text{ is zero} \\ q_i, & \text{otherwise.} \end{cases}$$

The cybersecurity loss of the CSP is equal to C when the attacker successfully compromises t VMs. Therefore, the cybersecurity gain of the attacker when attacking the CSP according to α is equal to

$$G(\alpha) = C \cdot \prod \widehat{Q} \prod \alpha \circ P, \tag{1}$$

where $\circ$ is the Hadamard matrix product and $\prod$ is taken over the non-zero entries of the corresponding matrix/vector. From this point forward, if we use the term "gain," we are referring to the attacker's gain from executing an attack. If we use the term "loss," we are referring to the CSP's loss from a successful cybersecurity attack.

3.2 Problem Definition

The CSP wants to ensure data privacy while using MPC in a cloud computation setting assuming malicious security. However, an attacker will try to compromise the data. We define a successful attack as any attack that can compromise t VMs and a worst-case attack is one that can compromise t VMs and maximize G across all such attacks. Since the attacker is rational and has perfect information, we assume the attacker will always execute a successful, worst-case attack. Given a CSP and an MPC protocol to perform, the CSP seeks to determine the cloud setup that minimizes the cybersecurity loss in the worst-case scenario. The CSP wants to choose a threshold t, security levels for the VMs (i.e. p_j), the number of MPC nodes (i.e. M), and allocation (i.e. β) such that L is minimal in the event of a successful, worst-case attack. In terms of our system model, we have the following:

$$\begin{aligned}
&\text{Minimize } L = \max_{\alpha \in A} \{G(\alpha)\} \\
&\text{Subject to} \\
&\quad G(\alpha) = C \cdot \prod \widehat{Q} \prod \alpha \circ P \\
&\quad \beta \in B \\
&\quad \alpha \in A \\
&\quad 1 \leq t \leq M \\
&\quad 2 \leq M = \sum_{i=1}^{N} m_i \\
&\quad 1 \leq N \\
&\quad 0 < p_j < 1 \\
&\quad 0 < q_i < 1
\end{aligned}$$

3.3 Solution

Before providing the solution, we make some observations that will help define the feasible regions for our parameters. Firstly, an attacker never targets an idle server. Note that the attacker's gain can be computed according to Equation (1). The relative gain for attacking H_N is zero since the server is idle. However, the probability of successfully attacking H_N is less than 1. Hence, even though the total gain is identical whether or not the final hypervisor is attacked, the attacker improves their chances of realizing this gain by not attacking H_N.

The threshold t should be as large as possible because the attacker has a higher chance of compromising t VMs than they do of compromising $t+1$ VMs. Note that the attacker's gain can be computed according to Equation 1. The probability of successfully attacking VM_{t+1} is less than 1. Therefore, the attacker's chances of successfully attacking all $t+1$ VMs are strictly smaller than the chances of successfully attacking t VMs. In particular, the CSP should always set $t = M$.

The provider should keep p_j, the probability of compromising VM_j, as close to zero as possible. If the attacker's chances of compromising a VM decrease, then their gain decreases (see Equation 1). In particular,

all p_j should be equal. To see this note that the gain from Equation 1 is larger if a single p_j is larger than the minimum p_j. Thus, the CSP should set all $p_j = p$, where p is as small as possible (i.e., making all VMs secure).

Finally, any allocation β with idle servers is never optimal in the worst-case scenario. Since the attacker will never attack idle servers, any allocation with an idle server provides no benefit to the CSP and can only have a greater loss. In particular, if $M < N$, then the CSP can improve their strategy by increasing M because any such allocation leaves at least one server idle. Without loss of generality, assume $N = M + 1$ and that server N is idle. Any attack strategy that involves attacking the hypervisor on server N reduces the security gain of the attacker who, being rational, would not choose to attack H_N.

Having chosen values for M, p_j, and t, the CSP only needs to choose an allocation strategy.

3.3.1 The Space of Attack Strategies

Since the CSP always chooses $t = M$, there is only one attack strategy that compromises the secret-shared data. Therefore, an optimal allocation may be determined by computing the gain the attacker receives against each allocation and choosing the one corresponding to the minimal gain.

THEOREM 3.1. *Any optimal allocation strategy is a Nash Equilibrium for this game.*

Proof. By definition, the CSP has no incentive to change strategies if they are using an optimal allocation. Since $t = M$, there is only one attack strategy that can compromise the secret-shared data. Therefore, the attacker cannot change strategies. Consequently, the allocation-attack strategy pair forms a NE. □

COROLLARY 3.2. *For a given M, N with $N \leq M$, a Nash Equilibrium always exists for this game.*

Proof. Because M and N are finite, there is always an allocation that gives the attacker minimal gain, i.e. less gain than the best attack against any other allocation (with the possibility of a tie). By the previous theorem, this optimal allocation is a NE. □

In the next section of the paper, we will enumerate the total number of optimal allocations. Although the total number of allocations is finite, it can be (depending on M and N) extremely large. Thus, it is beneficial to know if there is a formula for the number of optimal allocations.

3.3.2 The Space of Allocations and Stirling Numbers of the Second Kind

The total number of possible allocations is simply N^M, since each VM can be placed onto any of the N servers. However, the number of optimal allocations is generally much smaller than this. In particular, the number of optimal ways to allocate the VMs on the servers can be computed in terms of Stirling numbers of the second kind, which represents the number of ways to partition M objects into N non-empty, unlabeled subsets. Even when the servers are labeled, the number of allocations can be expressed in terms of these Striling numbers, which we denote $\left\{ {M \atop N} \right\}$ to indicate that we are placing M VMs onto N servers. Importantly, we have the following initial conditions:

$$\left\{ {M \atop M} \right\} = 1 \text{ for all } M \geq 1, \qquad \left\{ {M \atop 0} \right\} = 0 \text{ for all } M \geq 1, \qquad \text{and} \qquad \left\{ {0 \atop N} \right\} = 0 \text{ for all } N \geq 1,$$

The first condition says that there is only one way to place the VMs when $M = N$. The second says there is no way to place M VMs onto 0 servers. The third says there is no way to place 0 VMs onto N servers without leaving any servers idle.

The Stirling numbers of the second kind satisfy a recurrence relationship according to the following reasoning: in order to place $M+1$ VMs onto N servers, we may consider separately the case where VM_{M+1} shares a server, and the case where VM_{M+1} has a server to itself. In the former case, we have N choices for the placement of VM_{M+1} after we have placed the first M VMs onto N servers. In the second case, we must place the first M VMs onto $N - 1$ servers and place VM_{M+1} on the remaining server. This recurrence relation can be written mathematically as:

$$\left\{ {M+1 \atop N} \right\} = N \left\{ {M \atop N} \right\} + \left\{ {M \atop N-1} \right\}$$

and we can use this plus the initial conditions from before to write a formula for the Stirling numbers of the second kind:

$$\left\{ {M \atop N} \right\} = \frac{1}{N!} \sum_{i=0}^{N} (-1)^{N-i} \binom{N}{i} i^M = \sum_{i=0}^{N} \frac{(-1)^{N-i} i^M}{(N-i)!i!} \tag{2}$$

Next, we discuss different scenarios where servers and VMs can be identical or different in terms of their security levels. As we noted before, the CSP will always choose to make all the VMs equally secure and as secure as possible. Furthermore, the CSP will choose $t = M$ because this decreases the attacker's chances of compromising t VMs. Finally, we assume that $q_1, \ldots, q_N$ are constants. Therefore, we consider two cases: the case where all q_i and the case where all q_i are equal. In each scenario, we examine what the attacker's gain is for different allocations. Since $t = M$, there is only one attack strategy that can compromise the secret-shared data: to target all M VMs. Thus, the solution for this problem depends only upon the number of servers being utilized.

3.3.3 Indistinguishable Servers and Indistinguishable VMs

In this case, all the VMs have the same probability of being compromised, p, and all the hypervisors have the same probability of being compromised, q. Here Equation 1 shows that

$$G(\alpha) = C \cdot p^M \prod \widehat{Q}.$$

Since the attacker only has one strategy, the gain above is maximal. The loss is then minimized by any allocation that uses all N servers, in which case we have

$$L = C \cdot p^M \cdot q^N.$$

and there are $\left\{ {M \atop N} \right\}$ ways to allocate them. Each allocation gives the same worst-case cybersecurity loss. Thus, they are all optimal and, by Theorem 3.1, are all NE strategies.

3.3.4 Distinguishable Servers and Indistinguishable VMs

In this case, all the VMs have the same probability of being compromised, p, and the probability of compromising H_i is q_i. Here Equation 1 shows that

$$G(\alpha) = C \cdot p^M \prod \widehat{Q}.$$

Since the attacker only has one strategy, the gain above is maximal. The loss is then minimized by any allocation that uses all N servers, in which case we have

$$L = C \cdot p^M \prod_{i=1}^{N} q_i.$$

If the servers were indistinguishable, there would be $\left\{ {M \atop N} \right\}$ such allocations. However, this now undercounts by $N!$ since the servers are distinguishable, so the total number of allocations is

$$\left\{ {M \atop N} \right\} N!,$$

and they all give the same worst-case cybersecurity loss. Thus, they are all optimal and, by Theorem 3.1, are all NE strategies.

In the next section, we describe three algorithms for solving this problem. We show the performance of each algorithm and offer insights into how the algorithms compare to each other. The first two algorithms search the space of allocations to determine if/what the optimal allocations are. The third will construct the optimal allocations using the results in this section. As the number of servers increases, the runtime of all algorithms increases because the number of possible allocations is N^M and the number of optimal allocations is $\left\{ {M \atop N} \right\}$ (if servers are indistinguishable) or $\left\{ {M \atop N} \right\} N!$ (if servers are distinguishable). Similarly, if the number of VMs increases, the number of possible allocations and the number of optimal allocations also increase. However, the value of p and q_i do not affect the runtime because the number of possible allocations and optimal allocations do not depend on these values.

Approved for Public Release on 08 Mar 2024. Distribution Is Unlimited. Case Number: AFRL-2024-1317.

3.4 Evaluation

According to the theoretical discussion above, we may construct an optimal allocation by creating an $N \times M$ binary array with the property that (i) each column sum is 1 and (ii) each row sum is more than 0. The first condition guarantees that a VM is only allocated to a single server and the second condition guarantees that no server is idle. Since each such allocation is optimal, they all give the same loss in the event of a successful worst-case cybersecurity attack. For ease of computation, we deal with the case where servers are indistinguishable (and VMs are always indistinguishable in an optimal setup). Figure 1 below details the runtime results of a brute-force implementation for various M and N. Meanwhile, Figure 2 shows the response times of the MiniMax game algorithm. Finally, Figure 3 shows the response times of our MARS implementation.

Note that there are memory considerations that must be taken into account when implementing these algorithms, as the space of total allocations grows exponentially when M increases. In fact, the number of allocations grows quickly as N increases as well, but the rate of growth is determined by the current value of M, so it is generally a polynomial. Nevertheless, all experiments in this paper were performed in Python using the "yield" command, so as to perform operations on an allocation as it is computed, rather than requiring us to compute all allocations before performing any operations. In particular, we used a 12^{th} Gen. Intel(R) Core(TM) i7-12700 2.10GHz processor with 32.0 GB of installed RAM (31.7 GB usable) operating on a 64-bit Home Edition of Windows 11 with an x64-based processor. All results are displayed using a logarithmic scale, which captures the exponential behavior of the algorithm.

3.4.1 Evaluation of Brute Force Search

Algorithm 1 Finding all NE allocation strategies using the Brute Force Search

Require: $N \geq 1, M \geq 1$

$g_{\min} \leftarrow 1$
Allocs $\leftarrow$ [ ]
for $\beta \in B$ **do**
    $b \leftarrow \beta$
    $g \leftarrow G(b)$ ▷ Use Equation 1 to compute the gain for the attack that targets all VMs
    if $g = g_{\min}$ **then**
        Allocs.append(β)
    else if $g < g_{\min}$ **then**
        Allocs $\leftarrow \beta$ ▷ Overwrite the list of best allocations
        $g_{\min} \leftarrow g$ ▷ Overwrite the minimum gain

Figure 1 below displays the runtime results for the brute force search algorithm. It is apparent from the graphs that this search becomes infeasible rather quickly: even a modest case where we allocate 10 VMs onto 5 servers requires over 4 hours! The total number of possible allocations in this case is $|B| = 5^{10} = 9,765,625$.

For each allocation, we must compute the attacker's gain and compare it to the current calculated optimal gain, which adds additional time per allocation. Even though these computations are small, they compound quickly over the course of nearly 10 million allocations. By contrast, the total number of optimal allocations is only $\left\{ {10 \atop 5} \right\} = 42,525$ (or $\left\{ {10 \atop 5} \right\} 5! = 5,103,000$ if the servers are distinguishable). As a result, we conclude that the brute force search is an incredibly inefficient algorithm, especially given the small percentage of allocations that are optimal.

3.4.2 Evaluation of MiniMax

The next algorithm is MiniMax, which is a game theory algorithm that determines the existence of a NE strategy, however it may not find all NE strategies. It is known that this algorithm runs in polynomial time but, as we have seen already, the search space grows exponentially, making even this algorithm impractical for large M, N. Assuming we have indistinguishable servers (again, VMs will be indistinguishable because the CSP chooses them all to be equally secure), we can implement MiniMax to identify a strategy that minimizes the loss

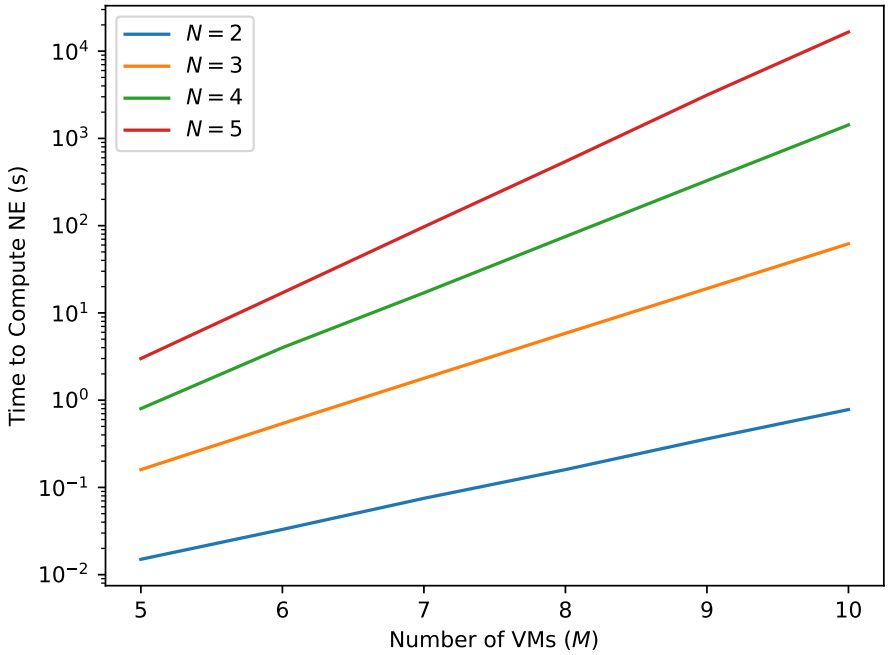

Figure 1: Runtime results of the Brute Force Search for finding Nash Equilibria

for a worst-case scenario. The algorithm evaluates the payoff of every possible strategy the player can take, and for each strategy, it considers the best possible counter-strategy the opponent could employ. In a zero-sum game, the opponent's best response corresponds to the player's worst-case outcome for that strategy. It then calculates the maximum loss (or minimum payoff) that the player could suffer for each strategy, given the opponent's best response. Finally, the algorithm selects the strategy that offers the best worst-case scenario—that is, the strategy that minimizes the maximum loss. This selected strategy is known as the player's minimax strategy. In this case, we compute the attacker's payoff matrix because MiniMax will find the smallest maximum in each row of the payoff matrix. If we use the CSP's payoff matrix, the gains would all be negative and MiniMax would not return the best strategy for the CSP, but would return the worst strategy for the CSP.

Algorithm 2 The Minimax Algorithm

```
function Minimax(payoff_matrix)
    Initialize max_losses to an empty list
    for each row in payoff_matrix payoff matrix of the attacker do
        Append the minimum of the row to max_losses
    end for
    min_max_loss ← maximum value in max_losses
    optimal_strategy_index ← index of min_max_loss in max_losses
    return optimal_strategy_index, min_max_loss
end function=0
```

MiniMax will still search the same set of strategies as the brute force algorithm. However, unlike brute force, it only needs to find the minimum of a list. Since MiniMax constructs the entries of the payoff matrix simultaneously, the runtime is from constructing the payoff matrix. Meanwhile, brute force must construct the gains individually and compare them to the current optimal gain. Moreover, it must store every optimal allocation, so it is required to check the entire list. The runtime results for MiniMax are shown in Figure 2

The case where $N = 5, M = 10$ is particularly interesting: Whereas the brute force algorithm took over 15,000 seconds (4+ hours), MiniMax took merely 324 seconds (5+ minutes). This speedup is significant, however, as

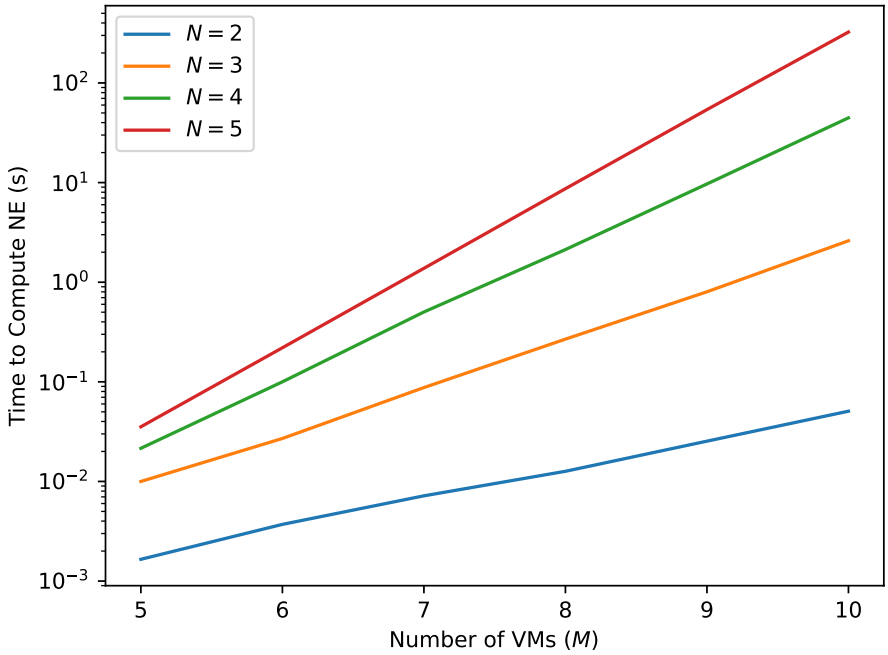

Figure 2: Runtime Results of The MiniMax Algorithm

we noted above, the search space grows exponentially, so examples with even one additional VM/server could double the runtime.

3.4.3 Evaluation of our MARS Implementation

Finally, we detail the runtime of our MARS implementation.

Algorithm 3 MARS Optimization

```
function STIRLING_ALLOCATIONS(M, N)                          ▷ Require M ≥ N ≥ 1
    for s in set_partitions([1,..., M], N) do                ▷ From the more_itertools package in Python3
        mat← numpy.zeros((N, M), dtype = int)                ▷ Initilize a 2D numpy array with all zeros
        for i in range(N) do
            for j in s[i] do
                mat[i][j − 1] = 1                            ▷ Update the entries so they reflect the allocation
            end for
        end for
        yield(mat)
    end for
end function
```

Our algorithm works by constructing the optimal allocations instead of searching for them. We know that any allocation using all N servers is optimal so, in order to construct an optimal allocation, we make sure that each row of the corresponding matrix has a non-zero entry. This saves time in the algorithm because we do not need to compute and compare the gain at each stage. Consequently, all of the runtime comes from constructing the allocation matrices.

All execution times for our MARS algorithm are under a minute in the cases considered. This is not always the case but, relative to the execution time of the other algorithms, it is clear that the mathematical approach using game theory is superior. The brute force algorithm reaches a one-minute execution time with $N = 3$ and $M = 10$, a case that MiniMax completed in 2.6 seconds, and MARS handled in 0.02 seconds. For reference, the

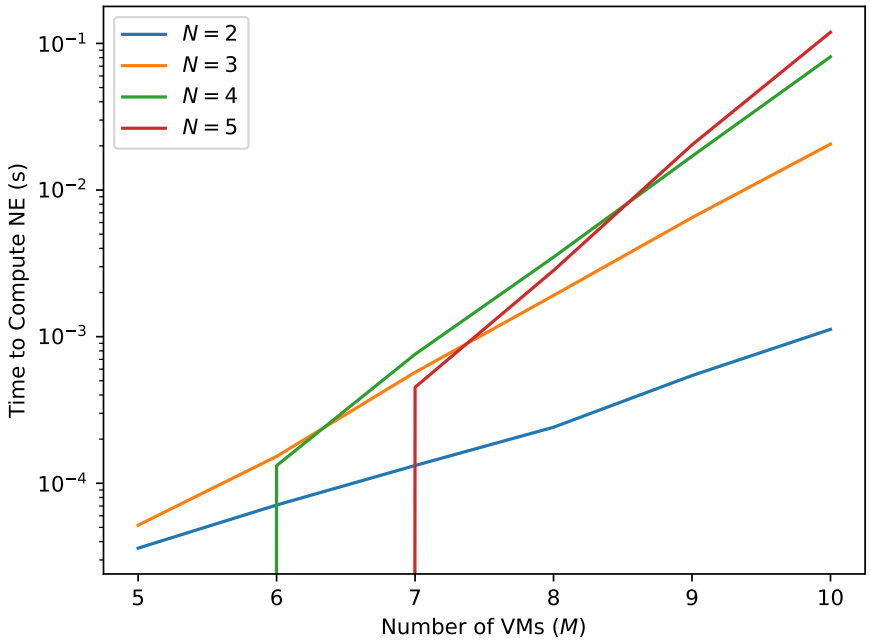

Figure 3: Timing Results From MARS Construction of Nash Equilibria

total number of allocations tried by the brute force algorithm is $3^{10} = 59,049$ and the total number of optimal allocations is $\left\{ {10 \atop 3} \right\} = 9,330$. By the time the MARS algorithm reaches a one-minute execution time—i.e. when $N = 7, M = 14$—the total number of optimal allocations is $49,329,280$, while the total number of possible allocations is $678,223,072,849$. MARS is also superior to MiniMax because it finds all NE allocations, whereas MiniMax only determines if there is an NE. Even if MiniMax was finding all optimal allocations, MARS outperforms MiniMax in runtime for all the experiments here by an order of magnitude.

Unlike the previous two algorithms, the exponential speedup is not immediately evident from the figure. In particular, the larger N initially has smaller runtimes than the smaller N, depending on M. This is because of the initial conditions for the number of allocations. For example, when $N = 4$ and $M = 5$, there are only 10 optimal allocations. Meanwhile, for $N = 2$ and $M = 5$, there are 15 optimal allocations. Therefore, the blue line is above the green line (in fact the green line is invisible because the runtime is 0.0 seconds) for $M = 5$. This changes as M increases and, eventually, the $N = 4$ (green) case runs more slowly than the $N = 2$ (blue) case. Similarly, the $N = 5$ case begins faster than all other cases, but quickly begins to run more slowly as the red line crosses the other lines on the graph.

4. PROBLEM II

It is often the case that a client will come to the CSP with specific parameters for an MPC protocol. Therefore, we wish to perform an analysis similar to the analysis we performed in the previous section, but we assume that

M, p_j and t are predetermined and the CSP only has control over β. In terms of our system model, we have:

$$\begin{aligned}
&\text{Minimize } L = \max_{\alpha \in A} \{G(\alpha)\} \\
&\text{Subject to} \\
&\quad G(\alpha) = C \cdot \prod \widehat{Q} \prod \alpha \circ P \\
&\quad \beta \in B \\
&\quad \alpha \in A \\
&\quad 1 \leq t < M \\
&\quad 2 \leq M = \sum_{i=1}^{N} m_i \\
&\quad 1 \leq N \\
&\quad 0 < q_i < 1 \\
&\quad 0 < p_j < 1
\end{aligned}$$

Consequently, the attacker gets to select from multiple attack strategies and the CSP must choose the allocation that gives the minimal gain to the attacker, who is going to use the strategy that maximizes their gain. As in the case of Problem I, the total number of possible allocations is N^M, so the brute force search will not offer a practical solution. The MiniMax algorithm will now take longer to run because the number of attack strategies increased. In Problem I, $t = M$ gave the attacker only one choice. In this problem, $t < M$ can be any number, depending on which MPC protocol the user chooses. For each t, there are

$$\binom{M}{t} = \frac{M!}{t!(M-t)!}$$

different attack strategies, so the payoff matrix is much larger. Finally, because servers may not be identical, it is not even apparent that an optimal solution should utilize every server. For illustration purposes, suppose that $t = 1$. Then, regardless of M and N, the CSP should allocate every VM to the most secure server, at which point the attacker will target that server and the least secure VM, thereby gaining $C \cdot p_M q_1$. Because of the added complexity of this problem, we save the analysis for future work. We can, however, describe the theoretical minimum loss in terms of the given parameters.

THEOREM 4.1. *Assume that $M \geq N$. Let $M = Nk + r$ where k is a positive integer and $1 \leq r \leq N - 1$. Let $1 \leq x \leq N$. If*

$$t > xk + r.$$

then there is an allocation strategy that forces the attacker to target at least $(x + 1)$ servers.

Proof. Distribute the VMs according to the following rules:

- Place $VM_1, \ldots, VM_k$ on server 1;
- While there are more than k VMs remaining, place the next (i.e. the lowest index) k VMs on the next (i.e. the lowest index) empty server;
- While there are VMs remaining, place the next VM (i.e. the lowest index) on the next (i.e., the lowest index) server with less than $k + 1$ VMs.

If M is a multiple of N, i.e. $r = 0$, then

$$\frac{Mx}{N} + r = xk,$$

and the maximum number of VMs the attacker can target by attacking only x servers is xk, because each server has exactly k VMs. However, this is less than t, so the attacker must target at least $(x + 1)$ servers.

Approved for Public Release on 08 Mar 2024. Distribution Is Unlimited. Case Number: AFRL-2024-1317.

The case $r \neq 0$ is similar. If $r \leq x$, then the maximum number of VMs that an attacker can compromise by targeting only x servers is

$$r(k+1) + (x-r)k = rk + r + xk - rk = xk + r$$

whereas, if $x < r$, the maximum number of VMs that an attacker can compromise by targeting x servers is

$$x(k+1) = xk + x < xk + r.$$

In both cases, this is strictly smaller than t. Thus, the attacker must target at least $(x+1)$ servers. □

There is always some value of x that makes the inequality $t > xk+r$ true because t is always at least 1. Therefore, we know that the theoretical minimum loss is achieved when the attacker is forced to target the x (which depends on t) strongest servers, and the t strongest VMs. Thus, $G_{\min} = C \cdot \prod_{j=1}^{t} p_j \prod_{i=1}^{x} q_i$.

This second research problem focuses on the security and resource allocation challenge within environments constrained by limited resources, specifically targeting VM allocation strategy as its core parameter. This problem diverges from the first, which deals with an unconstrained security scenario, by emphasizing the optimization of VM deployment in resource-limited settings to maintain security integrity. This nuanced approach seeks to balance effective defense mechanisms against potential threats while operating within the bounds of available computational and network resources. It may also be interesting in future work to develop algorithms that can dynamically adjust allocation in response to security challenges and resource constraints.

5. PROBLEM III

5.1 System model

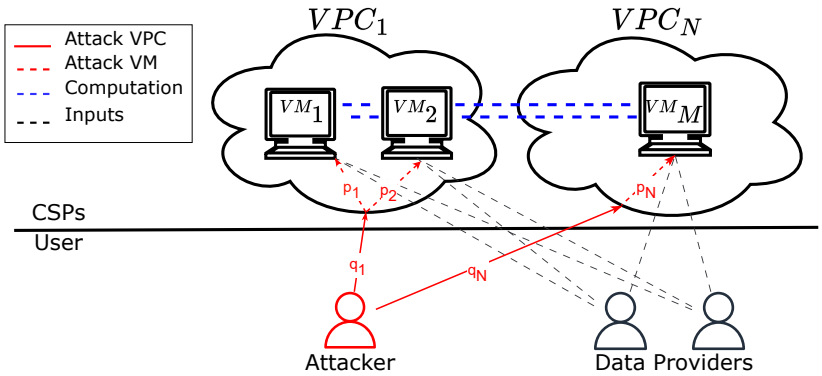

Figure 4: Problem III System Model overview

In this problem, we have modified our system model by considering additional factors that affect the computation Cost ($\mathcal{C}$) and also take into consideration Execution Time (ET). The new model is more representative of a real-life application of using MPC in ML training where a CSP trains on users' private data. Similarly to previous system models, if the privacy of the data is compromised, compensation will be required for the users. We expanded the system model to include the matrix multiplication optimization techniques presented in Hernandez et al.,[10] which is a common operation required in ML training. This work uses the Strassen algorithm to optimize matrix multiplication by lowering ET and $\mathcal{C}$ for each multiplication operation in a malicious dishonest majority network.

The system is divided into the User and CSP planes as shown in Figure 4. The User plane provides private data and requires services from the CSP plane. On the other hand, the CSP plane offers Virtual Private Clouds (VPCs) with allocated servers that run the MPC protocol. Our system assumes a broadcast channel where every server can communicate using the TCP/IP protocol. The private data is collected from users using Input Protocol,[25] which are secret-shared matrices used for matrix multiplications. Similarly to the previous system

model, the CSP creates VPCs to allocate servers using β. We also use $N \geq 2$, where N is the number of VPCs, and refer to each one using the following notation $\{N_0, N_1, \cdots, N_N\}$. These VPCs have an assigned value of $0 < q_i < 1$, which is the probability that N_i is successfully attacked. Communication between VPCs created in different CSPs and regions requires consideration of latency and bandwidth, which affects the cost. Furthermore, each VPC can host several servers, introducing additional resource allocation costs. We denote each server as $\{VM_0, VM_1, \cdots, VM_M\}$ where $M \geq 2$ represents the number of servers. Moreover, we use $0 < p_i < 1$ to represent the probability that an attacker will compromise VM_i.

We initially assume that the attacker must successfully compromise the VPC before attacking any VM, so the attacker receives no gain until this attack is complete. It is left to future work to consider a more dynamic case where the CSP can change allocations while the attacker is executing their attack. Given that the adversary gains solely from private data exposure, we assume that CSP should use $t = M$ for optimal privacy. SPDZ[26] protocol satisfies this condition, meaning the attacker must compromise t servers for a successful attack.

5.2 Problem Definition

The problem can be redefined based on the system model as a multi-objective optimization problem where the CSP needs to:

$$\begin{aligned} \text{Minimize} \quad & ET(\chi) \\ \text{Minimize} \quad & \mathcal{C}(\chi) \\ \text{Subject to} \quad & d \geq 0 \\ & \chi \in X \\ & t = M \end{aligned}$$

Where X is the set of all possible configurations determined by $M, N, \alpha, \beta, C, \{p_j\}$, and $\{q_i\}$ as in Problem 2. In addition to these configurations, we introduce other computation-related variables, which we will define as follows:

- z represents the size of the square matrices that are being multiplied, with each matrix having dimensions of $z \times z$.
- d is the depth and it is either zero when the naive matrix multiplication algorithm is used or non-zero when the Strassen's approach is used.
- s is the smallest size of z where the Strassen algorithm can be applied. It is defined by $s = \frac{z}{2^d} \quad s \geq 2$
- op is the number of required multiplication operations given d. It is defined by $op = 7^d \cdot s^3$
- $T_{\mathcal{L}}$ represents the time to perform a single multiplication operation given a specific latency $\mathcal{L}$.
- $\mathcal{C}_{VM}$ and $\mathcal{C}_{net}$ are hourly rates that apply to renting or allocating resources for computation, introduced later.

To define $ET(\chi)$, we use the following formula from the work of Hernandez et al.:[10]

$$ET(\chi) = (op \cdot T_L) + 7^d \mathcal{L} s \frac{M}{2}$$

In contrast to previous work, where the latency was the same for every allocation, the current approach defines latency (denoted by $\mathcal{L}$) as the maximum latency between any two participating VMs. Therefore, the $\mathcal{L}$ value represents the maximum latency between all VMs in the system. Since SPDZ is a synchronous protocol,

the value of $\mathcal{L}$ will be determined by the VM with the highest latency (assuming all communication happens in parallel).

Next, we define $\mathcal{C}(\chi)$ as follows:

$$\mathcal{C}(\chi) = RA(\chi) + L$$

where Resource Allocation (RA) cost refers to the total cost of using the network, CPU, RAM, and storage resources. In addition, L represents the security cost, which is the compensation cost in case of a successful attack, as defined in the Problem II. RA can be calculated using the following formula:

$$RA(\chi) = \underbrace{\mathcal{C}_{VM} \times ET(\chi)}_{\text{Hardware rate}} + \underbrace{\mathcal{C}_{net} \times TD(\chi)}_{\text{Network rate}}$$

Note that $\mathcal{C}_{VM}$ and $\mathcal{C}_{net}$ are flat average hourly rates the CSP charges to rent different VMs. These rates will vary depending on the location of the VPC and which CSP is hosting it. For example, if all the VMs are allocated in the same VPC, there will be no additional cost for communication. However, if they are located in different VPCs, there will be a communication fee:

$$\mathcal{C}_{net} = \begin{cases} 0, & \text{Same region or VPC} \\ > 0, & \text{Different region or VPC} \end{cases}$$

Moreover, $TD(\chi)$ represents the total data exchanged between VMs when the matrix multiplication is completed. It can be calculated using the formula $2f(M-1) \times op \times z$, where $f = 128$ bits is the field size used for the computation.

Finally, the attacker behavior will be similar to previously presented problems, and its influence will only affect the final value of $\mathcal{C}(\chi)$. Furthermore, the CSP will control the $RA(\chi)$ cost by selecting an allocation strategy, and this cost will be applied even if the $L = 0$ (i.e. the attack is not successful).

5.3 Solution without Using Game Theory

There are different approaches to solving this multi-objective optimization problem. We want to highlight two approaches from our previous work Hernandez et al.[10] that can address it. The first approach is brute force, which involves trying every possible configuration until the optimal solution is found. However, this approach, as we showed earlier in this work, can be time-consuming and requires substantial computing resources. Another approach is to use evolutionary algorithms such as NSGA-II, which do not provide the optimal solution but offers close solutions achieved in less time and with fewer computing resources. NSGA-II is a heuristic algorithm that can find almost optimal configurations.

5.3.1 Brute Force

Generally, for optimizations with a small variable space (i.e., few configurations), we iterate over all of them, looking for the best configuration, which we call brute force. This method always leads to the optimal configuration since it covers every possible value of χ. The approach involves using a utility function, denoted as $\mathcal{F}(\chi)$, to transform the multi-objective optimization problem into a single objective one. The $\mathcal{F}(\chi)$ function assigns a numerical value to each individual configuration, and the optimal configuration can be identified by finding the one with the minimum value of $\mathcal{F}(\chi)$. We do not recommend to solve the problem since it becomes increasingly slow when the possible configuration space increases.

5.3.2 NSGA-II

Another approach that offers a good solution in a relatively fast manner is using heuristic algorithms such as NSGA-II.[27] The improvements come at the expense of not finding the most optimal solution but solutions close to it. This evolutionary algorithm involves selecting a certain number of individuals (i.e., configurations or χ) in a population and the number of generations. At the start, the population has random configurations that will be improved in successive generations. The algorithm uses a fitness function $\mathcal{F}(\chi)$ to measure both objectives for each configuration, similar to the utility function from the Brute Force approach. After evaluating each individual in the population using $\mathcal{F}(\chi)$, only the highest-ranked solutions are retained, and a new set of individuals is created for the next generation. NSGA-II is highly scalable due to its ability to parallelize and avoid computing every possible configuration. However, there are two important reasons why using game theory would be a better approach to address our problem. Firstly, the solutions we get from NSGA-II may not be optimal. Secondly, we can improve the performance by identifying the Nash Equilibria and creating a custom algorithm for this problem.

5.4 Solution Using Game Theory

In the previous problem, the CSP was responsible for minimizing L by selecting the best strategies and parameters to run the training without losing money in compensation. When this problem is extended to a model where we also consider another cost that influences the CSP, such as the RA, we see that the net loss for the CSP is higher than the adversary. Not only do the adversary's actions impact potential cost, but so do the CSP's operational expenses. This complex dynamic establishes the foundation for a *non-zero-sum game* where the objectives of the CSP and the adversary differentiate, and their outcomes are not strictly opposite. Unlike our previous problems, the actions that result in a gain for the CSP will not be an identical loss for the attacker.

However, our CSP will manage the computational parameters required to provide the services (i.e., matrix multiplication in our case). At the same time, the attacker's strategy remains the same as in all other problems. We specifically address the case where $t = M$ since our ET formula is based on the SPDZ protocol.

6. CONCLUSION AND FUTURE WORK

In this ongoing research, we explore three problems and introduce several contributions: we model the cyber-security loss awareness problem as a two-player zero-sum game, identifying NE allocation strategies that yield optimal outcomes under malicious security and dishonest majority MPC settings. We propose a novel heuristic to efficiently find the NE in our game model, addressing the challenges posed by exponentially growing strategy spaces where traditional NE-search and brute-force algorithms become impractical and inefficient. Additionally, we model the security and constrained resources problem as a zero-sum game and solve it in the case where the VMs and servers are identical, and prove this problem is NP-hard. Future work includes incorporating security, resource costs, and response time and modeling this multi-objective optimization problem as a two-player, non-zero-sum game to find the optimal trade-offs between them across multiple VPCs in diverse geographic regions. Specifically, we aim to optimize ML operations across different CSPs (e.g., Azure, Google Cloud, and AWS) while accounting for potential security losses and execution time.

7. CONTRIBUTION

Mr. Artis Carter and Mr. Richard Hernandez contributed equally to this work, writing and executing much of the code for the various problems, plus writing significant portions of this paper. Drs. Homsi and Cosgrove functioned as advisors during the research and writing process, respectively.

ACKNOWLEDGMENTS

This research was supported partially by the Air Force Research Laboratory / Information Directorate (AFRL/RI) Internship program and The Virtual Institutes for Cyber and Electromagnetic Spectrum Research and Employ (VICEROY) program for summer 2023. We would like to offer a special thanks to Sonja Glumich, who manages the VICEROY program, for allowing us the opportunity to work with Mr. Carter. We also would like to thank Bobby Gigliotti and Walt Tirenin for their helpful comments in reviewing this work. The views and conclusions

contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory, Department of Defense, or the U.S. Government.

REFERENCES

[1] Chuang, F.-C., Tsai, Y.-N., Chow, Y.-N., Chuang, Y.-C., Huang, M.-C., Chuang, T.-W., and Chuang, T.-H., "Implementation of an e-learning platform in hybrid clouds," in [*2019 IEEE Eurasia Conference on Biomedical Engineering, Healthcare and Sustainability (ECBIOS)*], 91–94 (2019).

[2] Chahal, D., Mishra, M., Palepu, S., and Singhal, R., "Performance and cost comparison of cloud services for deep learning workload," in [*Companion of the ACM/SPEC International Conference on Performance Engineering*], *ICPE '21*, 49–55, Association for Computing Machinery, New York, NY, USA (2021).

[3] Qayyum, A., Ijaz, A., Usama, M., Iqbal, W., Qadir, J., Elkhatib, Y., and Al-Fuqaha, A., "Securing machine learning in the cloud: A systematic review of cloud machine learning security," *Frontiers in Big Data* **3** (12 2020).

[4] Sharma, S., Burtsev, A., and Mehrotra, S., "Advances in cryptography and secure hardware for data outsourcing," in [*2020 IEEE 36th International Conference on Data Engineering (ICDE)*], 1798–1801 (2020).

[5] Lindell, Y., "Secure multiparty computation (mpc)." Cryptology ePrint Archive, Paper 2020/300 (2020). https://eprint.iacr.org/2020/300.

[6] Cramer, R., Damgård, I., and Maurer, U., "General secure multi-party computation from any linear secret sharing scheme." Cryptology ePrint Archive, Paper 2000/037 (2000). https://eprint.iacr.org/2000/037.

[7] Araki, T., Furukawa, J., Lindell, Y., Nof, A., and Ohara, K., "High-throughput semi-honest secure three-party computation with an honest majority." Cryptology ePrint Archive, Paper 2016/768 (2016). https://eprint.iacr.org/2016/768.

[8] Goyal, V., Li, H., Ostrovsky, R., Polychroniadou, A., and Song, Y., "Atlas: Efficient and scalable mpc in the honest majority setting." Cryptology ePrint Archive, Paper 2021/833 (2021). https://eprint.iacr.org/2021/833.

[9] Chen, H., Kim, M., Razenshteyn, I., Rotaru, D., Song, Y., and Wagh, S., "Maliciously secure matrix multiplication with applications to private deep learning," in [*Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part III 26*], 31–59, Springer (2020).

[10] Hernandez, R., Akkaya, K., and Homsi, S., "Cost-based modeling and optimization of secure matrix multiplication in the cloud." ACM Conference on Data and Applications Security 2024 (2024). Under Review.

[11] Dalskov, A., Escudero, D., and Keller, M., "Secure evaluation of quantized neural networks," *arXiv preprint arXiv:1910.12435* (2019).

[12] Nash, J., "Non-cooperative games," *Annals of Mathematics* **54**(2), 286–295 (1951).

[13] Homsi, S., Quan, G., Wen, W., Chapparo-Baquero, G. A., and Njilla, L., "Game theoretic-based approaches for cybersecurity-aware virtual machine placement in public cloud clusters," in [*2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*], 272–281 (2019).

[14] Homsi, S., Quan, G., and Njilla, L., "Critical workload deployment in public clouds with guaranteed security levels and optimized resource usage and energy cost," in [*Proceedings of the Future Technologies Conference (FTC) 2018*], Arai, K., Bhatia, R., and Kapoor, S., eds., 240–260, Springer International Publishing, Cham (2019).

[15] Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T., "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii," in [*Parallel Problem Solving from Nature PPSN VI: 6th International Conference Paris, France, September 18–20, 2000 Proceedings 6*], 849–858, Springer (2000).

[16] Guariso, G. and Sangiorgio, M., "Improving the performance of multiobjective genetic algorithms: An elitism-based approach," *Information (Switzerland)* **12**, 587 (12 2020).

[17] Cui, S. and Homsi, S., "Perspective chapter: Deep reinforcement learning for co-resident attack mitigation in the cloud," in [*Artificial Intelligence Annual Volume 2022*], Fernandez, M. A. A. and Travieso-Gonzalez, C. M., eds., ch. 4, IntechOpen, Rijeka (2022).

Approved for Public Release on 08 Mar 2024. Distribution Is Unlimited. Case Number: AFRL-2024-1317.

[18] Goldreich, O., Micali, S., and Wigderson, A., "How to play any mental game or A completeness theorem for protocols with honest majority," in [*Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*], Aho, A. V., ed., 218–229, ACM (1987).

[19] Ben-Or, M., Goldwasser, S., and Wigderson, A., "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in [*Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*], *STOC '88*, 1–10, Association for Computing Machinery, New York, NY, USA (1988).

[20] Chaum, D., Crépeau, C., and Damgard, I., "Multiparty unconditionally secure protocols," in [*Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*], *STOC '88*, 11–19, Association for Computing Machinery, New York, NY, USA (1988).

[21] Cramer, R., Damgård, I., Dziembowski, S., Hirt, M., and Rabin, T., "Efficient multiparty computations secure against an adaptive adversary," in [*Advances in Cryptology — EUROCRYPT '99*], Stern, J., ed., 311–326, Springer Berlin Heidelberg, Berlin, Heidelberg (1999).

[22] Rabin, T. and Ben-Or, M., "Verifiable secret sharing and multiparty protocols with honest majority," in [*Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*], *STOC '89*, 73–85, Association for Computing Machinery, New York, NY, USA (1989).

[23] Yao, A. C.-C., "How to generate and exchange secrets," in [*27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*], 162–167 (1986).

[24] Ishai, Y., Prabhakaran, M., and Sahai, A., "Founding cryptography on oblivious transfer – efficiently," in [*Advances in Cryptology – CRYPTO 2008*], Wagner, D., ed., 572–591, Springer Berlin Heidelberg, Berlin, Heidelberg (2008).

[25] Damgård, I., Damgård, K., Nielsen, K., Nordholt, P. S., and Toft, T., "Confidential benchmarking based on multiparty computation." Cryptology ePrint Archive, Paper 2015/1006 (2015). https://eprint.iacr.org/2015/1006.

[26] Damgård, I., Pastro, V., Smart, N., and Zakarias, S., "Multiparty computation from somewhat homomorphic encryption," in [*Advances in Cryptology – CRYPTO 2012*], Safavi-Naini, R. and Canetti, R., eds., 643–662, Springer Berlin Heidelberg, Berlin, Heidelberg (2012).

[27] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation* **6**(2), 182–197 (2002).