A Multi-Objective Optimization Approach for Secure MPC in Cloud Environments

Richard Hernandez
Electrical and Computer Engineering
Florida International University
Miami, FL ,United States
rhern336@fiu.edu

Kemal Akkaya
School of Comp. and Inf. Sciences
Florida International University
Miami, FL,United States
kakkaya@fiu.edu

Soamar Homsi
The Information Assurance Branch
Air Force Research Laboratory
Rome, NY, United States
soamar.homsi@us.af.mil

***Abstract*—Cloud computing has transformed the deployment of Machine Learning (ML) applications, yet privacy and security challenges persist when processing sensitive data. Secure Multiparty Computation (MPC) protocols like SPDZ address these concerns but introduce computational overhead, particularly for matrix multiplication (MM) operations essential to deep learning (DL). This work optimizes secure MM performance over MPC by strategically allocating computational resources across Virtual Private Clouds (VPCs) and Virtual Machines (VMs) with varying attack probabilities. We formulate this as a Multi-Objective Optimization Problem (MOOP) that aims to minimize execution time and cost, where cost includes both computational expenses and potential liability from data breaches while reducing the risk of privacy violations. Using Game Theory methods and Pareto optimality, we develop three approaches: the first optimize execution time, the second prioritizes privacy, and the third balances both objectives. Our comprehensive evaluation proposes a deployment architecture for multiple cloud providers, compares allocation strategies under different VM and VPC distinguishability scenarios, and examines the impact of Strassen's algorithm depth on performance-security trade-offs. Implementation in the MP-SPDZ framework demonstrates that our approach significantly improves execution efficiency while maintaining strong privacy guarantees for sensitive ML workloads.**

***Index Terms*—Multiparty Computation, Cloud Computing, Cybersecurity, Game Theory, Non-cooperative, Zero-Sum, Multi-objective optimization, Machine Learning**

I. Introduction

Cloud computing has fundamentally transformed how organizations access and utilize computational resources. The on-demand services provided by Cloud Service Providers (CSPs) offer scalable, cost-effective solutions that integrate seamlessly across domains, particularly for Machine Learning (ML) applications. Despite these advantages, significant security and privacy challenges persist when processing sensitive data such as financial records or health information. When deploying ML applications, exposing training or inference data to CSPs creates unacceptable privacy risks, driving research into cryptographic protection techniques.

Two prominent approaches for privacy-preserving computation in the cloud are Secure Multiparty Computation (MPC) and Fully Homomorphic Encryption (FHE). While FHE offers strong theoretical guarantees, its high computational overhead renders it impractical for resource-intensive deep learning (DL) workloads. In contrast, MPC protocols offer a more balanced approach to privacy-preserving computation by enabling multiple parties to collaboratively compute functions over their inputs without revealing those inputs.

MPC protocols rely on secret sharing schemes to ensure data confidentiality while revealing only the final output. These protocols operate under various adversarial models, including honest-but-curious and malicious settings. This work addresses the malicious model, in which parties may deviate arbitrarily from the protocol. Among the prominent protocols in this space is SPDZ, a dishonest-majority solution that guarantees security-with-abort (halting if cheating is detected). While SPDZ offers robust privacy guarantees, it imposes significant computational and communication overhead, particularly for large-scale operations like matrix multiplication (MM), which is essential for DL training and inference.

To mitigate computational complexity, algorithms such as Strassen's method have been employed to reduce the arithmetic cost of secure MM. However, an often-overlooked factor is the allocation of computational resources across Virtual Private Clouds (VPCs) and Virtual Machines (VMs), which may face differing levels of adversarial risk. By strategically allocating resources, it is possible to reduce execution time and monetary cost while maintaining acceptable privacy guarantees. This trade-off becomes especially critical when deploying MPC-based ML workloads in multi-cloud settings.

In this work, we improve the performance of secure MM in MPC protocols like SPDZ by strategically allocating VMs across VPCs. We recognize that MM plays a crucial role in ML training and inference, and we must address the competing objectives of minimizing execution time and cost while assuring privacy. To tackle this, we formulate the problem as a Multi-Objective Optimization Problem (MOOP). Our model takes into account the distinct attack probabilities associated with each VPC and VM, and we solve this MOOP using Game Theory and the concept of Pareto optimality. We explore three complementary approaches: the first focuses on optimizing execution time, the second aims to ensure privacy, and the third balances both objectives through Pareto front analysis. Although we employ Strassen's algorithm to reduce arithmetic complexity, our main emphasis is on how decisions regarding resource allocation affect the performance and privacy of secure matrix multiplication under realistic attack scenarios.

The contributions of this work are threefold:

- *Multi-Objective Formulation:* We model the allocation of VMs across VPCs with distinct attack probabilities as a MOOP aiming to minimize execution time and cost while assuring privacy.
- *Enhanced MM Performance over MPC:* We introduce a strategic resource allocation scheme that leverages Strassen's algorithm to reduce computational overhead and improve MM performance in protocols like SPDZ.
- *Game-Theoretic and Pareto Approaches:* We solve the MOOP using Nash equilibrium concepts and Pareto optimality, developing allocation strategies that balance performance and security trade-offs for efficient and secure ML/DL workflows.

Our evaluation addresses four key research questions: (1) how to effectively deploy and integrate the proposed solution into existing cloud infrastructures; (2) how different VM and VPC allocation scenarios influence performance metrics; (3) how the three proposed strategies—Highest Privacy, Best Execution Time, and Optimal Tradeoff—compare in practice; and (4) how the depth of Strassen's algorithm affects performance-security trade-offs. The results demonstrate that combining strategic resource allocation with algorithmic optimization significantly enhances execution efficiency and reduces deployment costs, particularly for large-scale MM tasks in secure DL applications. The proposed approach provides a practical framework for deploying privacy-preserving ML in multi-cloud environments with heterogeneous security requirements.

The paper is organized as follows: Section II reviews related works. Section III introduces the SPDZ protocol and relevant game theory concepts. Section IV outlines our system model and formulates objectives for both the CSP and potential attackers. Section V presents our three solution approaches. Finally, Section VI details our experimental evaluation and results.

II. Related Works

The intersection of secure computation, resource allocation, and game theory has received increasing attention as cloud computing becomes ubiquitous for privacy-sensitive applications. We organize related work into three main categories: privacy-preserving protocols, resource allocation strategies, and game-theoretic approaches to cloud security.

A. Privacy-Preserving Protocols

Secure Multi-Party Computation has evolved significantly since Yao's seminal work on secure two-party computation [1]. Modern MPC protocols like SPDZ [2], MASCOT [3], and HIGHGEAR [4] have made substantial progress in efficiency and security guarantees. In particular, SPDZ provides security against malicious adversaries with dishonest majority, making it suitable for distributed cloud environments where trust cannot be guaranteed.

Several works have focused on optimizing MPC specifically for machine learning applications. Mohassel and Zhang [5] introduced SecureML, which optimizes two-party computation for various ML algorithms. Similarly, GAZELLE [6] combines homomorphic encryption with MPC to accelerate privacy-preserving neural network inference. However, these works primarily focus on algorithmic optimizations rather than resource allocation strategies across distributed environments.

B. Resource Allocation in Secure Computation

Resource allocation for secure computation presents unique challenges compared to traditional cloud resource management. Danezis et al. [7] analyzed the privacy-performance trade-offs in distributed systems but did not explore the game-theoretic aspects of resource allocation. Wu et al. [8] proposed privacy-aware task allocation in mobile edge computing, considering both computational efficiency and privacy requirements.

Most closely related to our work, Bautista et al. [9] introduced an MPC-as-a-Service framework that considers resource management across cloud providers. However, their approach does not incorporate attack probabilities or multi-objective optimization. Similarly, Chen et al. [10] optimized the offline phase of SPDZ using homomorphic encryption but did not address the strategic allocation of resources across potentially vulnerable infrastructure.

C. Game Theory in Cloud Security

Game theory provides powerful analytical tools for modeling strategic interactions between defenders and attackers in cloud environments. Kiennert et al. [11] surveyed game-theoretic approaches for cloud security, categorizing them into various model types including zero-sum, non-zero-sum, and Bayesian games. These models typically focus on general security measures rather than MPC-specific concerns.

Carter et al. [12] applied a game-theoretic approach to resource allocation for privacy preservation in cloud environments, modeling the interaction between CSPs and attackers as a security game. Their work forms the foundation for our security model, though we extend it to include execution time and computational cost in a multi-objective framework.

The application of Pareto optimality to balance security and performance objectives has been explored by Alam et al. [13] for cloud resource allocation, though not in the context of MPC. Similarly, Li et al. [14] proposed multi-objective optimization for task scheduling in clouds but did not consider privacy aspects or game-theoretic analysis.

While previous work has addressed various aspects of secure computation, resource allocation, and game theory independently, our approach makes several novel contributions. First, we uniquely combine game theory with multi-objective optimization to balance execution time, cost, and privacy in MPC deployments. This integration allows for a more holistic treatment of performance and security trade-offs. Second, unlike prior work that focuses primarily on algorithmic optimizations, we address the strategic placement of resources across VPCs with varying attack probabilities, introducing a more adversary-aware deployment strategy. Third, we provide concrete heuristics for resource allocation that are directly applicable to real-world MPC deployments, particularly for matrix multiplication operations essential to machine learning applications. By addressing these gaps, our approach enables more efficient and secure deployment of privacy-preserving machine learning in multi-cloud environments.

III. Preliminaries

This section introduces the foundational concepts and models underlying our approach: the SPDZ protocol for MPC, the security game framework, Nash equilibria and payoff matrices, and our system model.

A. SPDZ Protocol

The SPDZ protocol enables secure computation over private data under a malicious threat model. It operates in two phases: offline preprocessing and online computation.

In the offline phase, the participating parties $(P_1, P_2, \ldots, P_M)$ generate cryptographic materials necessary for the subsequent computation, including shared random values, multiplication triples, and zero-knowledge proofs.

The online phase encompasses input sharing, joint computation, and output reconstruction. Each party uses additive secret-sharing to distribute private data x into shares $x = x_1, x_2, \ldots, x_M$ and distributes them among all parties without revealing the original data. These shares possess homomorphic properties that enable computation directly on the shares.

For scalar multiplication in SPDZ, the protocol utilizes precomputed triples (a, b, c) generated during the offline phase via Beaver's technique [15], where $c = ab$. When each party P_i holds private input shares x_i and y_i (where $x = \sum_i x_i$ and $y = \sum_i y_i$), the multiplication proceeds as follows:

1) Each party computes and broadcasts $\epsilon_i = x_i - a_i$ and $\delta_i = y_i - b_i$
2) All parties calculate $\epsilon = \sum_i \epsilon_i$ and $\delta = \sum_i \delta_i$
3) Each party computes its share of the product: $z_i = c_i + \epsilon \cdot b_i + \delta \cdot a_i$
4) A designated party adds $\epsilon \cdot \delta$ to its share
5) The final result is constructed as $\sum_i z_i = ab + \epsilon b + \delta a + \epsilon\delta = xy$

Unlike honest majority protocols that can optimize matrix multiplication using dot products, dishonest majority protocols like SPDZ require communication for each scalar multiplication. This makes matrix multiplication computationally expensive in SPDZ, requiring n^3 scalar multiplications for $n \times n$ matrices.

B. The Security Game Model

In cloud environments, providers must secure computations against potential attackers while managing resource allocation efficiently. We adapt the security game model from Carter et al. [12] to our context as follows:

Users outsource data to CSPs for secure computation, and CSPs provide compensation in the event of data breaches. Each CSP manages a cluster of N cloud servers, with each server hosting VMs through virtualization technology. The probability of a server H_i being compromised is denoted as q_i $(0 < q_i < 1)$. If a server is compromised, all VMs on that server become vulnerable.

The attacker's objective is to gain unauthorized access to secret-shared data by compromising t VMs, where t represents the security threshold in the MPC protocol. The probability of any individual VM VM_j being compromised is denoted as p_j $(0 < p_j < 1)$.

The CSP employs an allocation strategy β, represented as an $N \times M$ matrix, to distribute M VMs across N servers, where:

$$M = m_1 + m_2 + \cdots + m_N$$

and m_i represents the number of VMs hosted on server H_i. Key parameters for the CSP include:

- N: number of cloud servers
- q_i: probability of server H_i being compromised
- M: total number of VMs
- p_j: probability of VM VM_j being compromised
- t: threshold number of compromised VMs required to breach security
- β: VM allocation strategy across servers
- C: financial compensation for data breaches

The attacker's strategy, denoted by α, is also represented as an $N \times M$ matrix. The expected gain from a successful attack is:

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

where $\circ$ represents the Hadamard product, $\prod$ denotes the product over non-zero entries, P is the matrix of VM compromise probabilities, and $\widehat{Q}$ is the vector of server attack probabilities based on the attacker's strategy.

Given that an attacker will always attempt the most effective attack by compromising exactly t VMs to maximize G, the CSP's challenge is to determine the optimal cloud configuration that minimizes its cybersecurity loss. This is formalized as:

$$\text{Minimize } L = \max_{\alpha \in A} \{G(\alpha)\}$$

where A represents the set of possible attack strategies.

In our work, we adapt this model by replacing hypervisors with Virtual Private Clouds (VPCs), as illustrated in Figure 1. Each VPC manages a set of VMs, with the compromise probability of a VPC being analogous to that of a hypervisor. This adaptation better reflects modern cloud architectures and enables more granular control over resource allocation and security management.

C. Nash Equilibria and Payoff Matrices

Nash's foundational work [16] established that every finite, n-player, non-cooperative game—including both zero-sum (ZSG) and non-zero-sum (NZSG) games—possesses at least one Nash equilibrium (NE), a state where no player

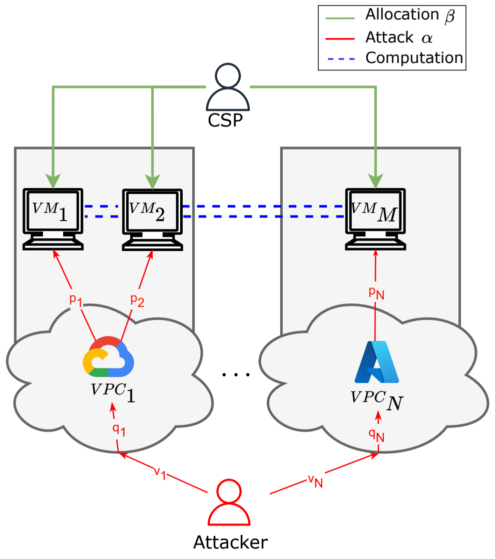


Figure 1: Game-theoretic model illustrating the strategic interaction between the attacker and CSP, with VPCs replacing hypervisors in our adaptation

can unilaterally improve their outcome by changing only their strategy.

In our context, we adopt a worst-case scenario perspective where the optimal strategy must withstand exploitation by a rational attacker. A CSP follows an optimal strategy when deviating from it offers no further benefit, assuming the attacker aims to maximize their gain. When both the CSP and the attacker act optimally, the system reaches an NE, though multiple such equilibria may exist.

Our game-theoretic model utilizes a payoff matrix, with CSP strategies represented by rows and attacker strategies by columns. Each matrix entry denotes the attacker's gain for a particular strategy combination. An NE can be identified by locating the row with the minimal maximum entry, representing the CSP's optimal strategy under worst-case assumptions.

While achieving an NE guarantees strategic stability, it does not necessarily yield the optimal individual outcome for either player. The following section focuses on minimizing the CSP's security loss while accounting for additional factors such as resource allocation efficiency and execution time.

D. System Model

Our system operates with a Management Server ($\mathcal{M}$) that coordinates outsourced computation as described by Bautista et al. [9]. This server is deployed as a third-party service responsible for deploying, coordinating, and optimizing MPC computations. The communication between $\mathcal{M}$ and both CSPs and clients involves configuration information that does not contain sensitive data used in the computation—including networking capabilities, protocol requirements, resource costs, hardware specifications, and other computation-specific parameters.

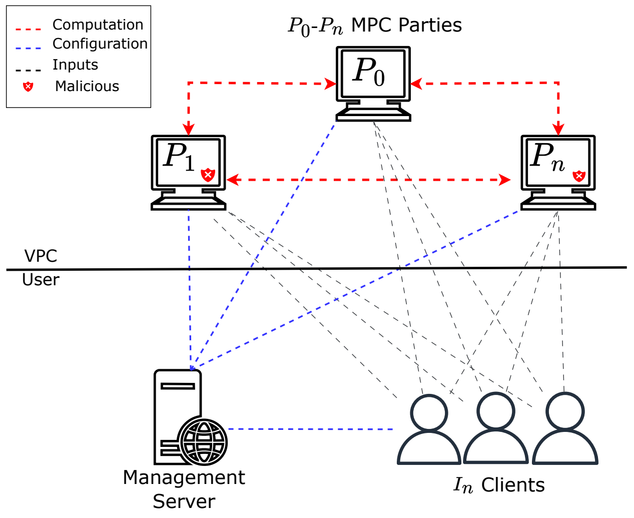


Figure 2: Communication architecture of the system model showing interactions between clients, cloud servers, and the management server

As illustrated in Figure 2, the system utilizes cloud servers $\{P_0, P_1, \ldots, P_M\}$, potentially hosted by different CSPs, and collaborating clients $\{I_0, I_1, \ldots, I_K\}$, where M and K represent the total number of MPC servers and clients, respectively. Each cloud server communicates with others using standard TCP/IP protocols. The inputs consist of pairs of square matrices, representing private training data in ML contexts.

We assume a dishonest majority network with up to $M - 1$ potentially malicious cloud servers and employ the SPDZ protocol to ensure privacy for all clients' data, using a field size $f = 128$. Clients share their data with servers using the input protocol outlined by Damgård et al. [17], which involves requesting offline data from MPC nodes and secret-sharing masked matrices.

IV. Problem Definition

In this section, we formulate the multi-objective optimization problem by defining the objectives of both the CSP and potential attackers. We present models for measuring execution time and total cost, enumerate possible resource allocation strategies, and outline the attacker's objectives.

A. CSP Objectives

We extend the traditional security game framework into a multi-objective optimization problem that considers both performance and security in MPC protocols with dishonest majority. To maximize security in our dishonest majority setting, we set the threshold t equal to the total number of parties M, ensuring that all parties must collude to breach the protocol's security.

Our primary objectives are to minimize both the execution time $ET(\chi)$ and the total cost $C(\chi)$ while maintaining privacy, where χ represents the set of optimization variables. The problem is formulated as:

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

Here, d is the depth parameter of the Strassen algorithm, which we control to balance computational complexity and communication overhead. By adjusting d, we can choose between the naive matrix multiplication algorithm ($d = 0$) and the Strassen algorithm at various recursion depths ($d \geq 1$):

$$d = \begin{cases} 0, & \text{Naive algorithm,} \\ \geq 1, & \text{Strassen algorithm.} \end{cases}$$

1) Execution Time Model: The execution time $ET(\chi)$ is defined as:

$$ET(\chi) = (op \cdot T_L) + \left(7^d \cdot L \cdot s \cdot \frac{M}{2}\right), \tag{1}$$

where:

- op is the number of basic operations required for the computation
- T_L is the time taken for a single multiplication operation
- L is the maximum network latency among all parties
- s is the size of the submatrices in the Strassen algorithm, defined as $s = \frac{A}{2^d}$, with the constraint $s \geq 2$
- A is the size of the matrices being multiplied

The number of operations op depends on the depth d of the Strassen algorithm and is given by:

$$op = 7^d \cdot s^3. \tag{2}$$

2) Network Latency Model: The maximum latency L is critical in synchronous protocols like SPDZ, where computation proceeds at the pace of the slowest communication link:

$$L = \max \{\text{Latency between party } i \text{ and party } j \mid 1 \leq i < j \leq M\}$$

For example, with three VMs, the latency matrix would be:

$$\begin{pmatrix} 0 & L_{12} & L_{13} \\ L_{21} & 0 & L_{23} \\ L_{31} & L_{32} & 0 \end{pmatrix}$$

where $L = \max\{L_{12}, L_{13}, L_{23}\}$.

3) Cost Model: The total cost $C(\chi)$ comprises two main components: the Resource Allocation Cost (RAC) and the Security Cost C_{sec}:

$$C(\chi) = \text{RAC} + C_{\text{sec}}. \tag{3}$$

The Resource Allocation Cost accounts for computational and networking resources:

$$\text{RAC} = C_M \cdot ET(\chi) + C_{\text{net}} \cdot TD(\chi), \tag{4}$$

where:

- C_M is the total hourly cost of all VMs, calculated as $C_M = \sum_{i=1}^{M} P_i$, with P_i being the hourly cost of VM i
- C_{net} is the networking cost, dependent on VM locations
- $TD(\chi)$ is the total data exchanged during computation

The networking cost C_{net} varies based on VM locations:

$$C_{\text{net}} = \begin{cases} 0, & \text{if VMs are in the same region or VPC,} \\ > 0, & \text{if VMs are in different regions or VPCs.} \end{cases}$$

The total data exchanged is given by:

$$TD(\chi) = 2f(M-1) \cdot op \cdot M, \tag{5}$$

where $f(M-1)$ represents the communication complexity per operation.

The Security Cost C_{sec} represents potential liabilities from data breaches:

$$C_{\text{sec}}(\alpha) = B \cdot \prod \widehat{Q} \prod (\alpha \circ P), \tag{6}$$

where B is the allocated budget for potential liabilities, and the equation inherits the allocation strategy variables from Carter et al. [12].

B. CSP Strategies

Due to the exponential number of possible configurations, we categorize strategies based on allocation patterns. This categorization helps manage the complexity of the optimization problem and provides insights into optimal allocation approaches for different scenarios.

1) Allocation Strategies: The values of M and N play a critical role in determining appropriate allocation strategies, as there are M^N possible allocations available to the CSP. These allocations create distinct configurations:

- When $M = N$, each server typically hosts exactly one VM
- When $M < N$, up to $N - M$ servers may remain idle
- When $M > N$, some servers must host multiple VMs

Analyzing these subcases enables tailored VM allocation strategies for different M and N configurations, improving overall system security against potential attacks, as illustrated in Figure 3.

2) Allocation Cases Based on Attack Probabilities: We further categorize allocation strategies based on the distinguishability of VMs and VPCs in terms of their attack probabilities.

Definition IV.1. *We classify CSP allocations into four cases:*

- ***CASE 1: Uniform Probabilities*** *- Both VMs and servers have identical attack probabilities: $\forall p_i = p$ and $\forall q_i = q$*
- ***CASE 2: Distinguishable Servers*** *- VMs have identical probabilities, but servers differ: $\forall p_i = p$ and $\forall q_i$ are distinguishable*

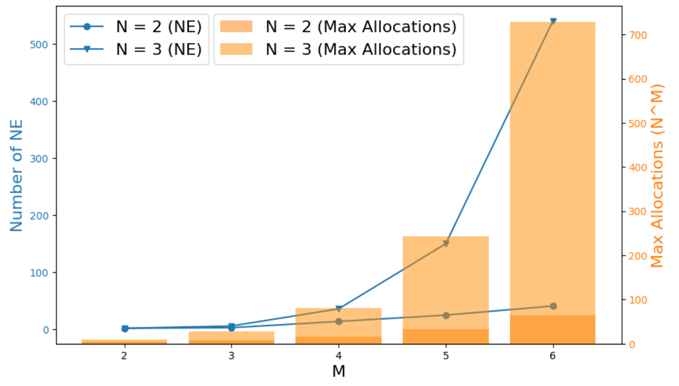


Figure 3: Relationship between the number of Nash Equilibria (NE) and maximum allocations as the number of VMs (M) increases, for different numbers of servers (N)

- ***CASE 3: Distinguishable VMs*** *- Servers have identical probabilities, but VMs differ:* $\forall p_i$ *are distinguishable and* $\forall q_i = q$
- ***CASE 4: Fully Distinguishable*** *- Both VMs and servers have distinct probabilities:* $\forall p_i$ *and* $\forall q_i$ *are distinguishable*

Each case requires unique optimization approaches and may yield different Nash Equilibria. For instance, in CASE 2, Figure 3 shows the relationship between the number of Nash Equilibria and the total number of allocations as M increases for $N = 2$ and $N = 3$.

C. Attacker Objectives and Strategies

In our model, we assume that the attacker's primary objective is to compromise enough VMs to breach the security of the MPC protocol.

Definition IV.2. *Given our use of the SPDZ protocol [2], we set* $t = M$*, indicating that an attacker must compromise all participating VMs to breach the protocol's security.*

This assumption reflects SPDZ's dishonest majority security model, where security is maintained unless all parties are compromised. The attacker gains nothing if the computation completes without a full compromise, making the attacker's success entirely dependent on the ability to compromise all VMs in the computation.

V. Approach

We formulate our problem as a NZSG, where the total losses and gains between the CSP and attacker are not balanced. In this formulation, the attacker's gain materializes only through a successful attack, which corresponds to an increase in the CSP's security cost (C_{sec}). To comprehensively analyze this game, we identify all possible allocation and attack strategies using payoff matrices. While this approach enables a systematic analysis of strategic interactions, we face a significant computational challenge: despite the game featuring a polynomial number of strategies when constructing the payoff matrix, the exponential growth in possible allocations as M and N increase makes exhaustive enumeration computationally infeasible.

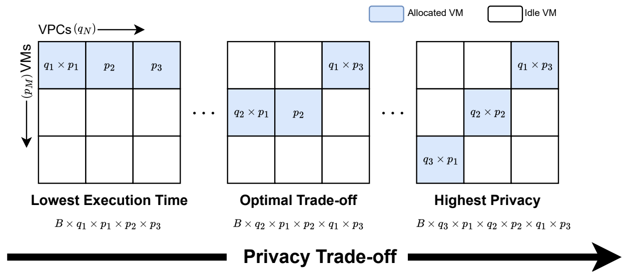


Figure 4: Overview of solution approaches across the security-performance spectrum, illustrating how different applications may prioritize execution time, privacy, or balanced trade-offs

To address this challenge, we divide our solution into three complementary approaches, each addressing a different optimization priority:

1) *Highest Privacy Solution:* Identifies allocation strategies that minimize potential security losses (C_{sec}) due to successful attacks, prioritizing data confidentiality above all other considerations.
2) *Lowest Execution Time Solution:* Focuses exclusively on minimizing execution time (ET), even when this might increase vulnerability to attacks, suitable for time-sensitive applications where computational speed is paramount.
3) *Optimal Tradeoff Solution:* Balances execution time and security considerations by identifying strategies that provide acceptable performance while maintaining adequate protection against attacks.

As illustrated in Figure 4, these three approaches cover the spectrum of possible optimization priorities in privacy-preserving computation. Each approach requires different assumptions and solution techniques. For the Highest Privacy approach, we simplify the problem by removing the execution time objective, transforming it into a ZSG game where the attacker's gain equals the CSP's loss in terms of security cost. The Lowest Execution Time solution focuses solely on minimizing latency and computational overhead without considering security implications. Finally, the Optimal Tradeoff approach addresses the full NZSG by combining game theory with Pareto optimality to identify solutions that balance both objectives effectively.

A. Highest Privacy Solution

This solution targets scenarios where data confidentiality is the paramount concern, such as applications involving healthcare data, government information, or proprietary business intelligence. In these contexts, the potential cost of a data breach far outweighs computational resource expenses. Since our primary objective is to minimize the likelihood of a successful attack, we reformulate the problem as a ZSG where the cost to the CSP is directly inverse to the gain for the attacker.

Building on the work of Carter et al. [12], which addresses security maximization when $M \geq N$, we extend the solution to cover all cases under Definition IV.2. We begin by analyzing the number of unique allocations under different configurations:

For $M = N$, the number of possible unique allocations is N. This is because any allocation that places exactly one VM per server is equivalent from a security perspective. Similarly, allocations that leave some servers empty only affect the overall security based on the number of servers hosting VMs. When $M > N$, a comparable number of unique allocations exists, as the placement of additional VMs on any already-occupied server does not affect security when q_i values are indistinguishable.

For $M < N$, the number of unique allocations is determined by M, as we can only allocate M VMs across the available N servers. Similar logic applies to CASE 3 from Definition IV.1, where server probabilities q are uniform but VM probabilities p_i vary.

Lemma 1. *In **CASE 1** and **CASE 3**, the Nash Equilibria (NE) of this game coincide with the optimal security strategy as follows:*

- *If $M \leq N$: Allocate exactly one VM per server, distributing VMs to maximize the number of servers an attacker must compromise.*
- *If $M > N$: Allocate one VM per server, and place remaining VMs on any of the servers, as all servers must be compromised regardless.*

For CASE 2 in Definition IV.1, where VMs have uniform probability but servers differ, we can compute the number of possible unique allocations using the combination formula $\binom{N+M-1}{N-1}$ or $\frac{(N+M-1)!}{(N-1)!M!}$. This corresponds to the classic combinatorial problem of placing M indistinguishable objects (VMs) into N distinguishable bins (servers) [18]. The indistinguishable nature of the VMs means that any allocation will be equivalent in terms of probability, affecting only which servers are targeted.

Lemma 2. *In **CASE 2**, the Nash Equilibria (NE) of this game yield the following optimal strategy for the CSP:*

- *If $M \geq N$: Allocate one VM to each server, and place any remaining VMs on the server with the lowest compromise probability (smallest q_i).*
- *If $M < N$: Allocate VMs one per server, prioritizing the servers with the lowest compromise probabilities, leaving the most vulnerable servers ($N - M$ servers with highest q_i values) without VMs.*

Finally, for CASE 4 (fully distinguishable VMs and servers), we have at most N^M possible unique allocations. However, given that the attacker must target all VMs under Definition IV.2, a similar approach to Lemma 2 applies, with the additional consideration of VM-specific compromise probabilities.

Algorithm 1 Generate VM Allocation Matrices for Nash Equilibria

```
Require: $n = |N|$ (number of servers), $m = |M|$ (number of VMs),
    $q_i$ (probabilities associated with servers), case $\in \{1, 2, 3, 4\}$
Ensure: allocations - a list of optimal VM allocations to servers
  allocations $\leftarrow \emptyset$
  if case $\notin \{1, 2, 3, 4\}$ then
      raise ValueError
  end if
  if $m < n$ then
      if case $= 1$ or case $= 3$ then
          allocations $\leftarrow \bigcup N_i$(arrangement)
            where arrangement $\in$ P$(N, M)$
      end if
      if case $= 2$ or case $= 4$ then
          $L_m \leftarrow$ Lowest$_m(N, q_i)$ ▷ Select $m$ servers with lowest $q_i$
          allocations $\leftarrow \bigcup N_i$(arrangement)
            where arrangement $\in$ P$(L_m, M)$
      end if
  else
      allocations $\leftarrow \bigcup$ S$(V, n)$      ▷ All servers get at least one VM
  end if
  return allocations
```

Algorithm 1 formalizes the approach to generating Nash Equilibrium allocations for each case, taking into account the specific constraints of Definition IV.2.

B. Lowest Execution Time Solution

This solution addresses scenarios where computational performance is the primary concern and security is a secondary consideration. Such situations arise in time-sensitive applications, real-time data processing systems, and computational tasks with strict service-level agreements (SLAs). Since execution time ET is significantly impacted by network latency in distributed MPC protocols, our objective is to identify allocation strategies that minimize the maximum communication delay between participating parties.

We demonstrate that to minimize execution time ($\min(ET)$), the optimal strategy places all VMs within a single VPC. This configuration eliminates inter-VPC communication delays, as intra-VPC communication typically has negligible network latency compared to communication across different VPCs. Under realistic assumptions of varying latencies between VPCs, the number of feasible solutions equals the number of available VPCs (N), with each solution corresponding to concentrating all VMs in one of the available VPCs.

Theorem 3. ***(Optimal VM Placement for Minimizing Execution Time)*** *Given a set of VMs $V = \{v_1, v_2, \ldots, v_M\}$ allocated across N VPCs, the optimal allocation strategy to minimize the total execution time is to place all VMs on a single VPC. Formally, let β_i denote the allocation strategy where all VMs are placed on VPC i, then:*

$$ET(V, \beta_i) \leq ET(V, \beta_j) \quad \forall j \neq i,$$

where $ET(V, \beta_i)$ represents the execution time for the set of VMs V when allocated according to strategy β_i, and VPC *i has the lowest internal communication latency among all VPCs.*

Proof Sketch. This allocation strategy minimizes the maximum latency L by eliminating inter-VPC communication delays, which significantly reduces the second term in Equation 1. As intra-VPC latency approaches zero, execution time becomes dominated solely by computational operations. The complete proof is provided in Appendix A. □

It is important to note that this solution represents one extreme of the optimization spectrum, where performance is maximized without explicit consideration of security implications. In practice, concentrating all VMs within a single VPC creates a single point of failure from a security perspective, as compromising that VPC would potentially compromise all VMs. This highlights the fundamental tension between performance and security that motivates our multi-objective approach.

The Lowest Execution Time solution serves as a reference point for performance-critical applications and establishes one boundary of the Pareto frontier we explore in the Optimal Tradeoff solution. By understanding this performance-optimal configuration, we can better evaluate the execution time penalties incurred when implementing more secure allocation strategies.

C. *Optimal Trade-off Solution*

The fundamental challenge in our problem arises from the need to simultaneously optimize two potentially conflicting objectives: execution time (ET) and cost (C). These objectives often exist in tension, where improving one leads to degradation in the other. Pareto optimality provides a principled framework for managing such trade-offs, allowing us to identify solutions where no objective can be improved without worsening another. This approach has been widely applied in fields ranging from economics to engineering optimization.

In our NZSG formulation, the CSP aims to optimize both ET and C while the attacker's goal remains fixed—to compromise all VMs regardless of the CSP's allocation strategy. This creates an asymmetric strategic interaction where the CSP's allocation decisions affect both objectives, while the attacker has a single dominant strategy due to the requirements of Definition IV.2.

The Pareto optimal solution integrates with our game theory framework by identifying allocation strategies where the CSP cannot improve either ET or C without worsening the other, given the attacker's fixed strategy. By analyzing the payoff matrix, we can identify the Pareto front—the set of non-dominated solutions that represent optimal trade-offs between execution time and security.

This approach yields allocation strategies that balance performance and security based on the CSP's risk tolerance and application requirements. The following theorem formalizes how Nash Equilibria can be determined in this context:

Theorem 4. ***(Best Response in Games with Fixed Opponent Strategy)*** *In a two-player game where one player's strategy is fixed (or the player has only one strategy), the Nash Equilibria are determined solely by the other player's strategies that are optimal responses to the fixed strategy. In such scenarios, any strategy by the adaptable player that cannot be unilaterally improved upon—given the fixed strategy of the opponent—constitutes a Nash Equilibrium.*

Proof Sketch. Since the attacker's strategy is fixed (attacking every VM), the CSP's problem reduces to finding their best response. The CSP's Pareto optimal strategies become Nash Equilibria in this game because the attacker cannot deviate from their fixed strategy. A detailed proof appears in Appendix B. □

The existence of Nash Equilibria in this context is guaranteed, as formalized in the following corollary:

Corollary 4.1. ***(Existence of Nash Equilibrium)*** *For any given finite M and N, at least one Nash Equilibrium exists in this game.*

Proof. Since both M and N are finite, the strategy space is finite, and by Nash's theorem, at least one Nash Equilibrium must exist. By Theorem 4, this equilibrium corresponds to a CSP optimal strategy. At this equilibrium, neither player can unilaterally improve their outcome by changing their strategy. □

1) Pareto Front Construction: To identify the Pareto optimal solutions, we first define the feasible solution space S based on our payoff matrix. For each strategy $s \in S$, we have values for each objective function $f_i(s) \in \{ET(s), C(s)\}$. We then identify the *ideal point* z^*, a hypothetical solution achieving the best possible value for each objective function simultaneously:

$$z^* = (f_1^{\min}, f_2^{\min})$$

where

$$f_i^{\min} = \min_{s \in S} f_i(s)$$

represents the lowest value of objective i across all feasible strategies.

Next, we calculate the distance $d(s, z^*)$ from each feasible strategy s to the ideal point z^* using the Euclidean distance:

$$d(s, z^*) = \sqrt{\sum_i [f_i(s) - f_i^{\min}]^2}$$

The set of *non-dominated solutions*—strategies for which no other strategy performs better in all objectives—constitutes the Pareto front. Formally, a strategy s is non-dominated if there is no other strategy $s' \in S$ such that:

$$f_i(s') \le f_i(s) \quad \text{for all } i$$

and

$$f_j(s') < f_j(s) \quad \text{for at least one } j.$$

2) Handling Scaling Challenges: Constructing the Pareto front presents several practical challenges. One major issue is the disparity in scale between ET and C, which can lead to uneven distributions of solutions. For instance, if the range of ET values $(\max(ET)-\min(ET))$ is significantly larger than the range of C values $(\max(C)-\min(C))$, the optimization process may be skewed toward minimizing execution time, thereby underrepresenting cost-effective configurations. Another complication arises from non-linear trade-offs. Specifically, variations in the Strassen depth parameter d can result in non-linear improvements in ET, which may inversely affect suboptimal configurations in complex and unpredictable ways.

To mitigate these issues, several techniques can be employed. First, normalization is used to standardize the scale of each objective, ensuring that neither dominates the optimization process. This is achieved through the transformation

$$f_i^{\text{norm}}(s) = \frac{f_i(s) - f_i^{\min}}{f_i^{\max} - f_i^{\min}},$$

which rescales each objective to the $[0, 1]$ interval. Second, the weighted sum method allows for prioritization by assigning weights to each normalized objective according to its relative importance. The overall score is computed as

$$F(s) = w_1 \cdot f_1^{\text{norm}}(s) + w_2 \cdot f_2^{\text{norm}}(s),$$

where $w_1 + w_2 = 1$ and each weight $w_i > 0$. Third, constraint handling can be applied by enforcing upper bounds on objective values to restrict the solution space to feasible regions. This is expressed as

$$f_i(s) \leq \epsilon_i,$$

where ϵ_i denotes the maximum acceptable value for objective i. Finally, for large and complex search spaces, evolutionary algorithms such as the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [19] can be used to approximate the Pareto front efficiently while preserving diversity among solutions.

Despite the effectiveness of these methods, the number of possible allocations grows exponentially with the number of VMs (M) and VPCs (N), rendering exhaustive evaluation impractical for large-scale deployments. In the next section, we introduce heuristic approaches designed to efficiently identify near-optimal allocation strategies without exhaustive enumeration.

D. Heuristics to Find an Optimal Strategy for CSP

The exponential growth in the number of possible allocations (M^N) makes computing Pareto optimality for every allocation computationally prohibitive as M and N increase. This challenge necessitates the development of efficient heuristics that can identify near-optimal strategies without exhaustive enumeration. Our heuristic approach leverages insights from the previous solutions to narrow the search space by eliminating allocations that are demonstrably suboptimal.

In our Optimal Trade-off solution, the Nash Equilibrium closest to the ideal point z^* (defined by the best possible values of ET and C) represents the most balanced allocation strategy. Using this principle, we develop a series of heuristics that efficiently identify promising allocation strategies for different scenarios.

1) Initial VM Allocation: The foundation of our heuristic approach is determining the optimal placement of the first VM, which establishes the starting point for subsequent allocations.

Theorem 5. ***(Optimal Initial VM Allocation)*** *For both objectives (ET and C), the optimal strategy places the first VM in the most secure VPC with the lowest communication latency to all other VPCs. This selection is justified by its proximity to the ideal point z^*, representing the optimal trade-off between ET and C. This principle applies to all cases defined in Definition IV.1.*

Proof Sketch. Both ET and C are minimization objectives with aligned optimization direction. Placing the first VM in the most secure VPC with lowest latency simultaneously reduces security cost and improves execution time. The full proof with case-specific analysis is provided in Appendix C. □

2) Incremental VM Allocation: After the initial allocation, we distribute the remaining VMs by recursively evaluating the marginal impact of each placement decision on both objectives. We define ΔET and ΔC as the incremental changes in execution time and cost, respectively, when transitioning from the current allocation to the next.

For each of the allocation cases defined in Definition IV.1, we have developed specific allocation strategies that build upon the initial VM placement approach. These allocation strategies (formalized as Corollaries 5.1–5.4 in Appendix D) follow a greedy approach where VMs are allocated incrementally, evaluating the impact on execution time and security objectives at each step. In general, when security concerns dominate ($\Delta C > \Delta ET$), VMs are distributed across different VPCs to enhance security at the cost of increased latency. Conversely, when performance is prioritized ($\Delta ET > \Delta C$), VMs are concentrated within fewer VPCs to minimize communication overhead.

The common pattern across all cases is the recursive comparison of trade-offs and the selection of allocations that minimize the distance to the ideal point z^*, thereby achieving near-optimal solutions that balance security and performance concerns.

3) Heuristic Algorithm: Algorithm 2 formalizes our heuristic approach, providing an efficient method for identifying near-optimal VM allocations across VPCs without exhaustive enumeration of all possibilities.

This algorithm implements a greedy approach that incrementally allocates VMs to VPCs while minimizing the distance to the ideal point z^*. By prioritizing VMs with lower compromise probabilities and evaluating each potential placement based on its impact on both objectives, the algorithm efficiently identifies allocation strategies that balance security and performance considerations.

The time complexity of this algorithm is $O(M \cdot N)$, making it tractable even for large numbers of VMs and VPCs. While it

Algorithm 2 Heuristic Algorithm for VM Allocation to VPCs

```
Require: $VMs$: List of virtual machines (ordered by ascending $p_i$)
Require: $VPCs$: List of virtual private clouds (ordered by ascending $q_i$ and lowest latency)
Require: $\Delta ET, \Delta C$: Normalized metrics for incremental execution time and cost
  Initialize $ET \leftarrow 0$, $C \leftarrow 0$    ▷ Total execution time and cost
  Identify ideal point $z^*$ based on minimum possible ET and C
  Allocate the first $VM_i$ to the $VPC_j$ with highest security and lowest latency    ▷ Apply Theorem 5
  for each remaining $VM_i \in VMs$ (in order) do
      $minDistance \leftarrow \infty$, $bestVPC \leftarrow \emptyset$
      for each $VPC_j \in VPCs$ do
          Compute $\Delta ET_{ij}$, $\Delta C_{ij}$ for placing $VM_i$ in $VPC_j$
          $newET \leftarrow ET + \Delta ET_{ij}$, $newC \leftarrow C + \Delta C_{ij}$
          $distance \leftarrow \sqrt{(newET - ET^*)^2 + (newC - C^*)^2}$    ▷ Distance to ideal point
          if $distance < minDistance$ then
              $minDistance \leftarrow distance$, $bestVPC \leftarrow VPC_j$
          end if
      end for
      Allocate $VM_i$ to $bestVPC$
      $ET \leftarrow ET + \Delta ET_{i,bestVPC}$, $C \leftarrow C + \Delta C_{i,bestVPC}$
  end for
  Output: Final allocation, $ET$, and $C$
```

does not guarantee global optimality in all cases (particularly for CASE 4 with non-strictly-ordered parameters), it provides high-quality solutions that closely approximate the true Pareto front at a fraction of the computational cost of exhaustive enumeration.

VI. Evaluation

In this section, we present a comprehensive evaluation of our proposed approaches for optimizing MPC through strategic resource allocation. To provide clear structure and focus, we organize our evaluation around the following research questions:

- **RQ1:** How can our approach be effectively deployed and integrated into existing cloud infrastructures?
- **RQ2:** How do different allocation cases (based on VM and VPC distinguishability) affect cost, execution time, and the number of Nash Equilibria?
- **RQ3:** How do the three proposed approaches (Highest Privacy, Lowest Execution Time, and Optimal Tradeoff) compare in real-world scenarios?
- **RQ4:** How does the depth of Strassen's algorithm impact performance and security trade-offs in different allocation scenarios?

Our evaluation employs both theoretical analysis and experimental implementation to address these questions, demonstrating the practical applicability and performance benefits of our approach.

A. RQ1: Deployment and Integration Assessment

In practical cloud environments, VMs and servers can exhibit different attack probabilities based on their geographical distribution and system configurations. These variations map directly to the cases defined in Definition IV.1.

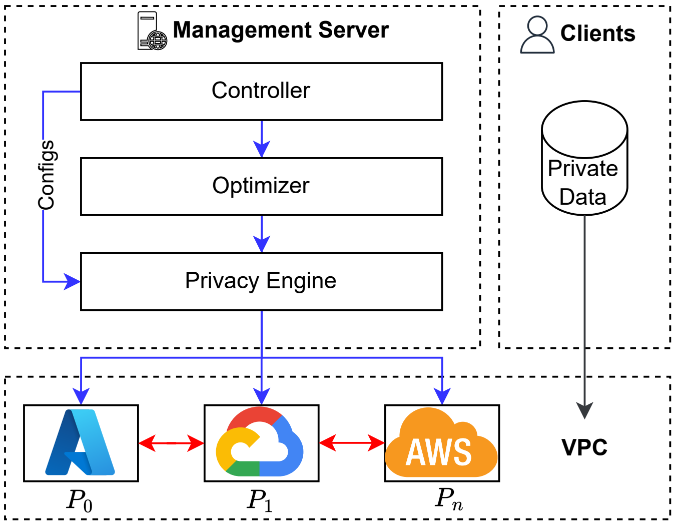


Figure 5: Deployment architecture showing the management server components and multi-cloud integration across AWS, Azure, and Google Cloud environments

1) Practical Deployment Scenarios: Different deployment scenarios naturally lead to different allocation cases:

- Deploying servers within the same region using identical Docker images creates an environment with indistinguishable VMs and servers (CASE 1).
- Deploying uniform Docker images across multiple regions results in distinguishable servers but indistinguishable VMs (CASE 2), as regional security policies and infrastructure may vary.
- Using heterogeneous VM images (e.g., different Linux distributions) from collaborating organizations creates environments with distinguishable VMs (CASE 3 or CASE 4).
- A healthcare provider provisioning VMs to execute MPC protocols with external agencies typically creates fully distinguishable VMs and servers (CASE 4).

2) Management Server Architecture: Figure 5 illustrates our proposed deployment architecture, comprising three core components:

- **Controller:** Orchestrates the computation process by aggregating configuration parameters from applications and selecting the appropriate optimization approach.
- **Optimizer:** Implements our proposed approaches (Highest Privacy, Lowest Execution Time, and Optimal Tradeoff) to identify optimal VM allocations based on the specific requirements.
- **Privacy Engine:** Provides the underlying MPC framework (e.g., MP-SPDZ) for secure computation, configuring the protocol based on the optimizer's allocation decisions.

This architecture can be deployed across multiple cloud providers (Azure, Google Cloud, AWS), with each cloud provider functioning as an independent MPC party within its respective VPC. This multi-cloud approach enhances security by distributing trust across different infrastructure providers.

B. RQ2: Numerical Analysis of Allocation Cases

1) CSP Allocations under Different Configurations: We examined cost (C) under various configurations of VPCs and VMs, as shown in Table I. Our analysis focused on how different allocation strategies impact total possible allocations (TA), Nash equilibria (NE), and cost (C) across two scenarios: $M < N$ (fewer VMs than VPCs) and $M \geq N$ (equal or more VMs than VPCs), using naive matrix multiplication ($d = 0$).

Table I: Allocation analysis results comparing total allocations (TA), Nash Equilibria (NE), and cost (C) across different cases for $M < N$ and $M \geq N$ configurations

Case	**TA**	**NE**	**Cost (C)**
$M < N$ **(N = 4, M = 3)**			
CASE 1	64	6	238.35
CASE 2	64	6	85.62
CASE 3	64	6	189.06
CASE 4	64	6	69.84
$M \geq N$ **(N = 4, M = 4)**			
CASE 1	256	4	118.27
CASE 2	256	1	59.68
CASE 3	256	4	80.62
CASE 4	256	1	44.62

We used the following parameters for our evaluation: $T_L = 0.0001$, $C_M = 1 \times 10^{-11}$, $C_{\text{net}} = 1 \times 10^{-10}$, and VPC latencies ranging from 1 to 3 milliseconds. The attack probabilities varied by case: CASE 1 used uniform probabilities of $q_i = [0.33, 0.33, 0.33]$ and $p_i = [0.33, 0.33, 0.33]$; CASE 2 used $q_i = [0.17, 0.33, 0.5]$ and uniform $p_i = [0.33, 0.33, 0.33]$; CASE 3 used uniform $q_i = [0.33, 0.33, 0.33]$ and varying $p_i = [0.17, 0.33, 0.5]$; and CASE 4 used both $q_i = [0.17, 0.33, 0.5]$ and $p_i = [0.17, 0.33, 0.5]$.

Our results demonstrate several key observations. CASE 4 consistently yields the lowest cost (C), which can be attributed to the optimizer's ability to leverage both VM and VPC distinguishability for optimal resource placement. Additionally, when $M > N$, the cost generally decreases and the number of Nash equilibria may reduce due to the increased likelihood of repeated configurations. Finally, CASE 2 and CASE 4 tend to exhibit lower costs than CASE 1 and CASE 3, as the differences in q_i values allow for more effective security-driven optimization.

C. RQ3: Comparison of Solution Approaches

We analyzed our three solution approaches using parameters $M = 3$ and $N = 3$ for CASE 4, which represents the most realistic scenario with fully distinguishable VMs and VPCs. We configured a latency matrix with incremental latency (starting at 1ms for the first VPC and increasing by 1ms for each subsequent VPC).

Table II: Impact of Strassen's algorithm depth on Nash Equilibria (NE) count, execution time (ET), and cost (C) for CASE 4

	Naive	**Strassen (Depth d)**			
	$d = 0$	$d = 1$	$d = 2$	$d = 3$	$d = 4$
$M < N$					
ET (s)	104.08	94.43	96.73	101.11	174.90
C ($)	3520.12	3084.24	2699.41	2385.13	2090.49
NE	6	6	6	1	1
$M > N$					
ET (s)	103.14	92.63	90.46	112.08	213.32
C ($)	5278.14	4620.35	4043.11	3538.02	3096.06
NE	7	1	1	1	1

Figure 6a shows the execution time (ET) and cost (C) values for all 27 possible strategies ($N^M = 3^3 = 27$). Among these strategies, only 13 form the Pareto front, with one solution (highlighted in red in Figure 6b) representing the optimal trade-off closest to the ideal point z^*. This optimal solution has $ET = 769.34$ seconds and $C = 3995.29$.

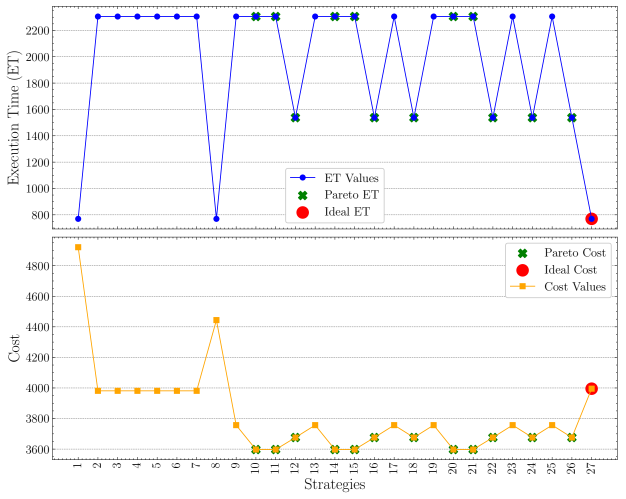
The corresponding latency matrix for this optimal allocation is:

$$\begin{array}{c|ccc} & \text{VM}_1 & \text{VM}_2 & \text{VM}_3 \\ \hline \text{VM}_1 & 0 & 1 & 1 \\ \text{VM}_2 & 1 & 0 & 1 \\ \text{VM}_3 & 1 & 1 & 0 \end{array}$$

The Lowest Execution Time solution is represented by strategies 1, 8, and 27 in Figure 6a, where all VMs are placed within the same VPC. While these strategies minimize latency due to intra-VPC communication, the associated security cost varies due to CASE 4's differentiated server probabilities. Among them, Strategy 27 is particularly notable for placing all VMs in the most secure VPC, thereby achieving the lowest execution time with relatively lower cost.

In contrast, the highest privacy is achieved by strategies located at the far end of the Pareto front in Figure 6b, specifically strategies 10, 11, 14, 15, 20, and 21. These strategies distribute VMs across all available VPCs, thereby maximizing security. However, this comes at the expense of increased latency, reaching up to 3ms, which significantly impacts execution time.

Finally, the optimal trade-off solution, highlighted in red in Figure 6b, achieves a balance between execution time and security cost. This solution offers superior performance compared to the highest privacy strategies while maintaining stronger security guarantees than the lowest execution time solution. As such, it represents the most efficient compromise between latency and privacy for CASE 4 under the chosen configuration.


(a) Execution time and cost values for all 27 possible strategies with $M = 3$ and $N = 3$

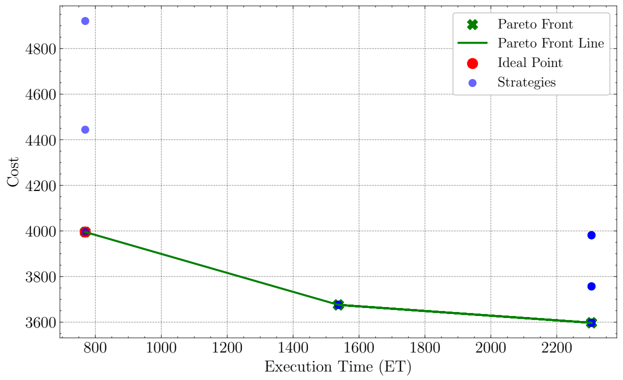

(b) Pareto front showing the trade-off between execution time and cost, with the optimal solution highlighted in red

Figure 6: Comparison of allocation strategies and the resulting Pareto front for CASE 4 with $M = 3$ and $N = 3$

D. RQ4: Impact of Strassen Algorithm Depth

We investigated the impact of varying the Strassen algorithm's recursion depth (d) on performance and security trade-offs for CASE 4, which represents the most realistic scenario with fully distinguishable VMs and VPCs. We increased the matrix size to 1024×1024 to better showcase Strassen's performance characteristics with larger matrices. Table II shows our results for both $M < N$ and $M > N$ configurations.

As the recursion depth increases, the number of Nash Equilibria consistently decreases. This reduction arises because deeper levels of recursion introduce greater communication overhead, which in turn makes latency a more critical factor in the optimization process. Consequently, the optimization tends to converge to fewer stable configurations.

The execution time exhibits a non-monotonic trend. It initially decreases at moderate depths ($d = 1$ and $d = 2$) due to a reduction in the number of arithmetic operations. However, beyond this point, execution time increases again at higher depths ($d = 3$ and $d = 4$), where communication costs dominate and diminish the gains from reduced computation.

In contrast to execution time, the cost metric (C) decreases monotonically with increasing depth. This consistent reduction reflects the optimizer's shift toward more secure allocations as latency penalties increase. At higher depths, the optimization framework places greater emphasis on minimizing exposure to risk, leading to lower overall costs.

Additionally, the effect of system configuration is noteworthy. In the $M > N$ setting, better execution times are observed at intermediate depths ($d = 1$ and $d = 2$), despite the increased computational load from a higher number of VMs. This improvement, however, comes with elevated security costs, indicating a trade-off that must be managed based on application requirements.

These results demonstrate that the optimal Strassen depth depends on the specific matrix size, network conditions, and security requirements. For the 1024×1024 matrices in our experiments, $d = 2$ appears to offer the best trade-off for the $M > N$ configuration, while $d = 1$ is optimal for $M < N$.

E. Discussion

Our evaluation demonstrates that strategic resource allocation significantly impacts both the performance and security of MPC protocols. The key findings are:

- **RQ1:** Our approach can be effectively deployed across multiple cloud providers using a management server architecture with Controller, Optimizer, and Privacy Engine components.
- **RQ2:** The distinguishability of VMs and VPCs (CASE 4) enables more effective optimization, resulting in lower costs and potentially fewer Nash Equilibria.
- **RQ3:** The Optimal Tradeoff approach successfully balances execution time and security considerations, offering a middle ground between the Highest Privacy and Lowest Execution Time solutions.
- **RQ4:** Strassen's algorithm depth introduces a non-monotonic relationship with execution time but monotonically decreases security costs, with optimal depth dependent on matrix size and network conditions.

These findings confirm that our multi-objective optimization approach effectively addresses the inherent trade-offs in MPC, providing practitioners with flexible strategies to balance performance and security based on their specific requirements.

VII. Conclusion

This work comprehensively explores optimizing secure matrix multiplication over MPC protocols, such as SPDZ, with a strategic focus on resource allocation. By formulating the problem as a Multi-Objective Optimization Problem (MOOP), we've demonstrated how to minimize execution time and costs while ensuring robust privacy in cloud computing environments. The use of game theory and Pareto optimality frameworks enabled the identification of optimal strategies that balance execution efficiency and security objectives.

Our experimental results confirm the efficacy of the proposed solutions, particularly the ability to adapt resource allocation across VPCs and VMs with varying attack probabilities. These findings underscore the critical impact of strategic allocation in mitigating privacy risks while optimizing computational efficiency in large-scale applications. Moreover, the integration of Strassen's algorithm highlights its potential to further reduce computational overhead, though requiring careful trade-off considerations to balance increased communication costs.

The proposed approaches, validated through both theoretical modeling and empirical evaluation, provide practical implementations for privacy-sensitive applications, including healthcare, finance, and government sectors. This work contributes to enhancing the robustness and efficiency of cloud-based MPC frameworks, with potential extensions to include evolving threat models, dynamic resource environments, and integration with other cryptographic techniques.

VIII. Acknowledgment

This work is supported by the AFRL/RI Internship Program for Summer 2023. 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 or the U.S. Government.

IX. References

[1] A. C. Yao, "Protocols for secure computations," in *23rd annual symposium on foundations of computer science (sfcs 1982)*. IEEE, 1982, pp. 160–164.

[2] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty Computation from Somewhat Homomorphic Encryption," in *Advances in Cryptology – CRYPTO 2012*, ser. Lecture Notes in Computer Science, R. Safavi-Naini and R. Canetti, Eds. Berlin, Heidelberg: Springer, 2012, pp. 643–662.

[3] M. Keller, E. Orsini, and P. Scholl, "MASCOT: Faster Malicious Arithmetic Secure Computation with Oblivious Transfer," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, Oct. 2016, pp. 830–842. [Online]. Available: https://doi.org/10.1145/2976749.2978357

[4] M. Keller, V. Pastro, and D. Rotaru, "Overdrive: Making SPDZ Great Again," in *Advances in Cryptology – EUROCRYPT 2018*, J. B. Nielsen and V. Rijmen, Eds. Cham: Springer International Publishing, 2018, pp. 158–189.

[5] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 19–38.

[6] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "{GAZELLE}: A low latency framework for secure neural network inference," in *27th USENIX security symposium (USENIX security 18)*, 2018, pp. 1651–1669.

[7] G. Danezis, "The least privacy-damaging centralised traffic data retention architecture," in *Security Protocols XVII*, B. Christianson, J. A. Malcolm, V. Matyáš, and M. Roe, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 93–110.

[8] Z. Wang, J. Hu, R. Lv, J. Wei, Q. Wang, D. Yang, and H. Qi, "Personalized privacy-preserving task allocation for mobile crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 6, pp. 1330–1341, 2018.

[9] O. G. Bautista and K. Akkaya, "MPC-as-a-Service: A Customizable Management Protocol for Running Multi-party Computation on IoT Devices," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, May 2023, pp. 1–6, iSSN: 2374-9709. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10154349

[10] H. Chen, M. Kim, I. Razenshteyn, D. Rotaru, Y. Song, and S. Wagh, "Maliciously Secure Matrix Multiplication with Applications to Private Deep Learning," in *Advances in Cryptology – ASIACRYPT 2020*, ser. Lecture Notes in Computer Science, S. Moriai and H. Wang, Eds. Cham: Springer International Publishing, 2020, pp. 31–59.

[11] C. Kiennert, Z. Ismail, H. Debar, and J. Leneutre, "A survey on game-theoretic approaches for intrusion detection and response optimization," *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, pp. 1–31, 2018.

[12] A. Carter, R. Hernandez, S. Homsi, and G. M. Cosgrove, "Security games with malicious adversaries in the clouds: status update," in *Assurance and Security for AI-enabled Systems*, vol. 13054. SPIE, Jun. 2024, pp. 197–215. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/13054/130540O/Security-games-with-malicious-adversaries-in-the-clouds–status/10.1117/12.3014000.full

[13] A. B. Alam, Z. M. Fadlullah, and S. Choudhury, "A resource allocation model based on trust evaluation in multi-cloud environments," *IEEE Access*, vol. 9, pp. 105 577–105 587, 2021.

[14] W. Li, Q. Fan, F. Dang, Y. Jiang, H. Wang, S. Li, and X. Zhang, "Multi-objective optimization of a task-scheduling algorithm for a secure cloud," *Information*, vol. 13, no. 2, p. 92, 2022.

[15] D. Beaver, "Efficient Multiparty Protocols Using Circuit Randomization," in *Advances in Cryptology — CRYPTO '91*, ser. Lecture Notes in Computer Science, J. Feigenbaum, Ed. Berlin, Heidelberg: Springer, 1992, pp. 420–432.

[16] J. Nash, "Non-cooperative games," *Annals of Mathematics*, vol. 54, no. 2, pp. 286–295, 1951.

[17] I. Damgård, K. Damgård, K. Nielsen, P. S. Nordholt, and T. Toft, "Confidential Benchmarking based on Multiparty Computation," *Cryptology ePrint Archive*, 2015. [Online]. Available: https://eprint.iacr.org/2015/1006

[18] W. Feller *et al.*, "An introduction to probability theory and its applications," 1971.

[19] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002. [Online]. Available: http://ieeexplore.ieee.org/document/996017/

Appendix

This appendix provides detailed proofs for the theorems and corollaries presented in the main text.

A. Optimal VM Placement for Minimizing Execution Time

Theorem 3 (Optimal VM Placement for Minimizing Execution Time): This theorem establishes that for a set of VMs allocated across multiple VPCs, placing all VMs on a single VPC with the lowest internal communication latency minimizes the total execution time.

Proof. This allocation strategy minimizes the maximum latency L by eliminating inter-VPC communication delays. When all VMs are co-located within the same VPC, we have $L \approx 0$ for their interactions, since intra-VPC communication typically occurs over high-speed local networks with minimal latency. This significantly reduces the second term in our execution time shown in Equation 1.

As the latency L approaches zero, the execution time is dominated solely by the computational component $op \cdot T_L$, which represents the time required for the actual arithmetic operations. Any allocation that distributes VMs across multiple VPCs would introduce non-zero inter-VPC latencies, increasing the overall execution time.

When selecting among the N possible single-VPC allocations, the optimal choice is the VPC with the lowest internal processing latency (if VPCs have different internal performance characteristics) or the lowest operational cost (if performance is uniform across VPCs). This results in exactly N possible solutions, with one optimal solution corresponding to the VPC with the best performance characteristics. □

B. Best Response in Games with Fixed Opponent Strategy

Theorem 4 (Best Response in Games with Fixed Opponent Strategy): This theorem states that in a two-player game where one player's strategy is fixed, Nash Equilibria are determined solely by the other player's strategies that are optimal responses to the fixed strategy.

Proof. Given that the attacker's strategy is fixed (they always attack every VM), the CSP's problem reduces to finding their best response to this fixed strategy. The CSP cannot influence the attacker's actions but can adjust their own strategies to optimize ET and C.

By focusing on Pareto optimal solutions, the CSP identifies strategies where any unilateral change would not lead to a better outcome in both objectives. Since the attacker cannot change their strategy, the CSP's Pareto optimal strategies become Nash Equilibria in this game. □

C. Optimal Initial VM Allocation

Theorem 5 (Optimal Initial VM Allocation): This theorem establishes that for both objectives (ET and C), the optimal strategy places the first VM in the most secure VPC with the lowest communication latency to all other VPCs.

Proof. Both ET and C are minimization objectives, creating alignment in their initial optimization direction. Allocating the first VM to the VPC with the highest security (lowest q_i) reduces the security cost component, while selecting a VPC with minimal latency improves execution time. This dual optimization is possible because reducing security parameters (q and p) always results in a lower overall cost, and minimizing network latency directly mitigates the exponential growth of ET.

The application of this principle varies across the cases in Definition IV.1:

- **CASE 1** (Uniform Probabilities): When all VPCs have identical security probabilities, the minimal latency becomes the determining factor. If multiple VPCs have identical latencies, any can be selected arbitrarily.
- **CASE 2** (Distinguishable Servers): When VPCs have different security probabilities, allocation prioritizes VPCs with the lowest q_i values. If multiple VPCs have identical q_i values, selection is made based on minimal latency.
- **CASE 3 and CASE 4** (VMs with varying probabilities): The VM with the lowest compromise probability (p_i) is placed in the most secure VPC to optimize the security cost. If security and latency considerations conflict (e.g., the most secure VPC has the highest latency), security considerations take precedence because VMs placed in the same VPC inherently benefit from reduced intra-VPC latency.

In all cases, this initial allocation establishes a foundation that simultaneously addresses both security and performance objectives. □

D. Optimal Allocation Strategies for Different Cases

This section presents the optimal allocation strategies for each of the four cases defined in Definition IV.1.

1) Optimal Allocation for CASE 1:

Corollary 5.1. ***(Optimal Allocation for CASE 1)*** *Given uniform probabilities across all VPCs and VMs ($\forall p_i = p$ and $\forall q_i = q$), and following the initial allocation from Theorem 5, the optimal allocation strategy is:*

- *For $M \geq N$: Compare ΔET and ΔC for each potential allocation. If $\Delta C > \Delta ET$ (normalized), place the next VM in a different VPC with the lowest latency; otherwise, keep it in the same VPC as the initial VM.*
- *For $M < N$: Apply the same comparative approach, resulting in $M - N$ VPCs remaining idle.*

Proof. This greedy approach is effective because VMs allocated incrementally contribute independently to the aggregate metrics. After the initial allocation, each subsequent VM placement affects the objectives in one of two ways: either ET increases while C decreases (as derived from Lemma 1), or ET decreases while C increases (from Theorem 3). By recursively comparing these trade-offs and selecting the allocation that minimizes distance to z^*, we achieve a near-optimal solution. □

2) *Optimal Allocation for CASE 2:*

Corollary 5.2. ***(Optimal Allocation for CASE 2)*** *When VMs have uniform probabilities but VPCs are distinguishable ($\forall p_i = p$ and $\forall q_i$ vary), the optimal allocation strategy is:*

- *For $M \leq N$: If $\Delta C > \Delta ET$ (normalized), place the next VM in the VPC with the next lowest q_i and acceptable latency; if $\Delta ET > \Delta C$, place it in the same VPC as the initial VM.*
- *For $M > N$: Once all VPCs have at least one VM, place remaining VMs in the VPC with the lowest q_i to minimize security risk.*

Proof. Similar to Corollary 5.1, the linear contribution of each VM to ET and C enables effective greedy allocation. The key difference is that varying q_i values across VPCs create different security risk profiles. This typically causes ΔC to increase more gradually when allocating to secure VPCs, resulting in a tendency to place more VMs in VPCs with minimal q_i values, even at the cost of some latency penalty. □

3) *Optimal Allocation for CASE 3:*

Corollary 5.3. ***(Optimal Allocation for CASE 3)*** *When VMs have different compromise probabilities but VPCs are uniform ($\forall p_i$ vary and $\forall q_i = q$), the optimal allocation follows the approach in Corollary 5.1, but begins with the VM having the lowest p_i and proceeds in ascending order of p_i.*

Proof. Since all p_i values are included in the attacker's gain calculation (per Definition IV.2), but all q_i values are identical, the allocation strategy focuses on placing VMs with lower p_i values first. The comparative logic for ΔC and ΔET remains consistent with Corollary 5.1, as moving VMs to additional VPCs still improves security at the cost of increased latency. □

4) *Optimal Allocation for CASE 4:*

Corollary 5.4. ***(Optimal Allocation for CASE 4)*** *When both VMs and VPCs have varying probabilities ($\forall p_i$ and $\forall q_i$ are distinguishable), an optimal allocation can be found by:*

- *Starting with the VM that has the lowest p_i value and placing it according to Theorem 5*
- *Proceeding with VMs in ascending order of p_i*
- *Applying the allocation logic from Corollary 5.2*

This approach is guaranteed optimal only when all p_i values, all q_i values, and all inter-VPC latencies are strictly ordered (no ties). Otherwise, a more comprehensive analysis may be required.

Proof. The distinguishing feature of CASE 4 is that both VMs and VPCs have varying security parameters, leading to more complex interactions between allocation decisions and security outcomes. Starting with the VM that has the lowest p_i value ensures the most security-critical component is allocated optimally. The incremental allocation following the principles from Corollary 5.2 then balances the security and performance trade-offs for subsequent VMs. When strict ordering exists for all parameters, this approach identifies at least one Nash Equilibrium. □