An SMC Framework for Privacy-Preserving Clinical Prediction Across Hospitals

Clément Moréna, Dominique Huang, Élia Mounier-Poulat

***Abstract*—Medical institutions often need to collaborate on patient care while maintaining patient privacy and complying with regulations like HIPAA. We present a Secure Multi-party Computation (SMC) framework that enables multiple healthcare facilities to collectively compute medical predictions without revealing their sensitive patient data. Performance analysis demonstrates that our system scales effectively with up to 50 participating institutions, though communication overhead increases exponentially with the number of parties.**

Contribution: We all contributed equally to the project, but more precisely: **Clément Moréna:** Secret Sharing, Trusted Party, Communication Performance; **Dominique Huang:** Expressions, SMC Performance; **Élia Mounier-Poulat:** Beaver Triplets, Custom Application.

I. Introduction

In today's data-driven environment, institutions frequently need to collaborate by processing sensitive information while preserving privacy. Secure Multi-Party Computation (SMC) is a cryptographic technique that enables multiple parties to jointly compute arithmetic circuit over their private inputs without exposing the underlying data. This framework supports essential operations such as the addition and multiplication of both secret and scalar values, and it can be seamlessly integrated into statistical model inference for diverse parties. Although SMC and related privacy-enhancing technologies provide robust security guarantees, they also introduce performance overhead due to the inherent complexity of secure computations. This report presents a comprehensive performance analysis of an SMC framework, focusing on both server cost and operation-level efficiency, and explores a real-world application scenario that demonstrates its practical utility in the clinical world.

II. Threat model

Our approach operates under an honest-but-curious threat model, where all participating entities are assumed to follow the protocol correctly, yet may attempt to extract additional information from the exchanged messages. We assume the existence of a trusted third party to facilitate secure communication and that no collusion occurs between parties. We also assume that all communications are encrypted. The primary security objectives of our implementation are to ensure the privacy of individual inputs and to guarantee the correctness of the computed results.

III. Implementation details

Our implementation is based on the open-source framework provided in [1] and developed in Python 3.9. It incorporates additive secret sharing, with each secret value divided into shares and distributed among all participating parties. To ensure robust security and arithmetic operations, we operate within a finite field defined by a large prime number.

Each party locally computes its share of the result based on the given arithmetic expression (e.g., addition or multiplication of secret and scalar values). After local computation, parties publish their results, which are then combined to reconstruct the final output. Only multiplication is less intuitive and uses the notion of Beaver Triplet. We implemented unit tests for each component, including secret sharing, arithmetic operations, and reconstruction, and conducted performance tests to evaluate both computation and communication costs, with the results presented in Section 4. The experiments were repeated between 5 and 30 times, depending on the running time for statistical reliability. Note that these tests were performed on a CPU only; GPU evaluations for larger-scale experiments (e.g., 1,000 or 5,000 operations) were not conducted, as we hypothesized that for higher numbers of parties and/or operations, the hardware specifications of the device used might impact performance. However, we stated that most of the computation time is due to the communication protocol which is not impacted by the hardware.

IV. Performance evaluation

A. Network impact

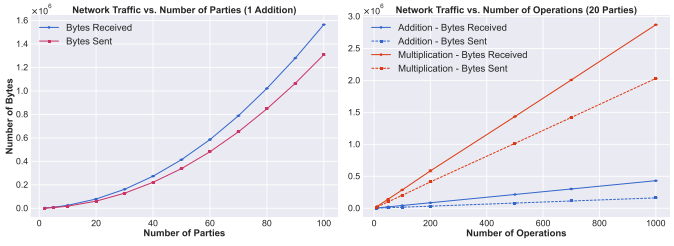

Fig. 1: Communication Cost Analysis

We measured the total number of bytes sent and received by the trusted party. The responses from the trusted party were significantly lower than the requests as a result of the overhead of the `HTTP GET` headers. Our analysis revealed that increasing the number of participants leads to an exponential increase in network costs, whereas increasing the number of operations

results in only a linear increase, regardless of the type of operation. In particular, we did not observe differences in network cost between secret and scalar operations. Additional Note: the STD on the plots are very small, so the results are quite precise and reproducible.

B. Operation impact

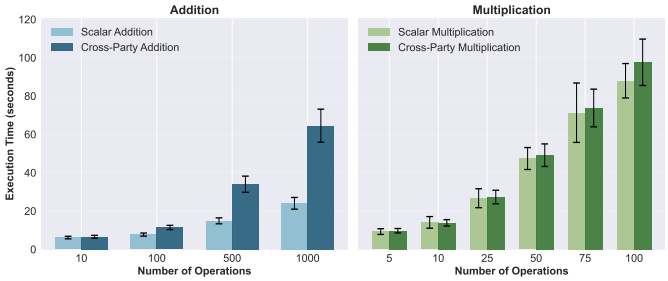

Fig. 2: Execution Time vs. Number of Operations (20 Parties)

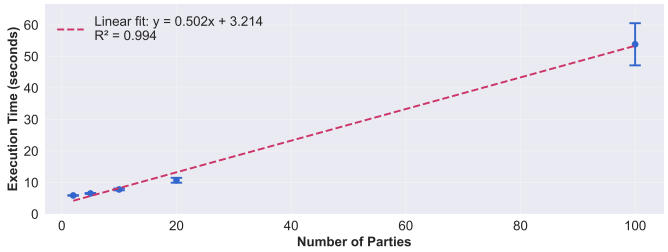

Fig. 3: Execution Time vs. Number of Parties (100 Scalar Additions)

We also measured how the number of operations and parties affect execution time. As indicated by the Beaver triplets implementation, performing 100 multiplications incurred roughly the same cost as 1,000 additions, demonstrating that multiplications are significantly more computationally expensive. In addition, increasing the number of parties resulted in a linear increase in execution time, which suggests that the performance impact scales predictably as the system grows.

V. Application

Our SMC framework can be used for critical healthcare applications, such as determining whether patients require transfer to specialized care facilities based on distributed medical records. In modern healthcare systems, patient data is typically fragmented across institutions—for example, laboratory results at one hospital, symptom documentation at another, and medical history records at a third. Our engine can compute a *risk score* that measures the urgency of a patient's case based on these records while keeping each hospital's sensitive information confidential.

A *risk score* could be modeled by the following circuit:

$$[\text{Risk}] = [w_0] + [\sum_i w_i X_i] + [\sum_{i<j} w_{ij}(X_i X_j)]$$

where w_0 is a bias term, X_i represents a patient feature contributed by one of the hospitals, w_i and w_{ij} are weight coefficients, and $X_i X_j$ denotes the interaction between two features. The final computation could later be compared to a predefined threshold to evaluate how urgent the patient's situation is.

A. Privacy Leakage Analysis

While our implementation prevents the direct exposure of input data to other participating institutions, several privacy vulnerabilities remain even within our honest-but-curious, no-collusion adversarial model.

First, our system is susceptible to output inference attacks. The final risk score may leak sensitive information, particularly when computations are performed repeatedly on similar patient cases with slight variations. For instance, a hospital might deduce another institution's input values by observing changes in the output when its own inputs remain constant.

Moreover, the inclusion of interaction terms $X_i X_j$, while clinically valuable, could expose additional information compared to linear terms alone, as these terms may reveal relationships between inputs from different parties.

B. Mitigation Strategies

To address these privacy vulnerabilities, one possible mitigation strategy is to incorporate differential privacy by adding calibrated noise to the output, which helps obscure slight variations in the risk score and makes it more difficult for adversaries to infer individual inputs.

Alternatively, instead of sharing precise risk scores, each party could map its computed value into a predefined categorical range (e.g., low, medium, high), and the final decision could be determined by majority vote or by averaging these categorical outcomes.

Another approach is to perform local aggregation, at each institution; i.e computing their piece of the model before sharing it for the overall statistical model, reducing the granularity of the shared information.

VI. Conclusion

We have demonstrated a practical SMC framework for healthcare applications that balances security and efficiency. The system's communication and computation scale linearly with the number of operations, but the number of parties increases server performance requirements exponentially. Future work will focus on optimizing communication, mitigating privacy leaks through differential privacy, and extending functionality to support more complex operations.

References

[1] S. EPFL, "Cs-523-public," https://github.com/spring-epfl/CS-523-public, accessed: February 28 2025.