Practical Privacy-Preserving Gaussian Process Regression via Secret Sharing (Supplementary Material)

Jinglong Luo[1,2] **Yehong Zhang**[2] **Jiaqi Zhang**[2] **Shuang Qin**[2] **Hui Wang**[2] **Yue Yu**[2] **Zenglin Xu**[1,2]

[1]Harbin Institute of Technology, Shenzhen, China
[2]Peng Cheng Laboratory, Shenzhen, China

A CORRECTNESS AND SECURITY ANALYSIS OF PP-EXP

A.1 PROOF OF THEOREM 1

In terms of algorithm correctness, we need to consider the problem of overflow or underflow errors during the execution of Algorithm 2. First, the PP-Exp algorithm involves computations on two algebraic structures, namely, the ring of integers $\mathcal{Z}_L$ modulo L and the fixed-point set $\mathcal{Q}_{<\mathcal{Z}_L,l_f>}$. Specifically, the operations on shares are performed on $\mathcal{Z}_L$ and others are performed on $\mathcal{Q}_{<\mathcal{Z}_L,l_f>}$.

Underflow: The PP-Exp has the risk of underflow when calculating $e^{-\check{r}}$ and $e^{\check{d}}$ in Line 5 and Line 13 of Algorithm 2. The minimum value of $e^{-\check{r}}$ and $e^{\check{d}}$ is $e^{u_{min}-\check{r}_{max}}$. To guarantee that PP-Exp will not overflow during computation, we need ensure that $e^{u_{min}-\check{r}_{max}}$ can be expressed as a fixed-point number with precision l_f. This is equivalent to $e^{u_{min}-\check{r}_{max}} \cdot 2^{l_f} \geq 1$. Therefore, $l_f \geq (\check{r}_{max} - u_{min}) \log_2^e$ is needed to ensure that PP-Exp does not underflow during the calculation.

Overflow: PP-Exp has the potential to overflow when computing $[e^u]_j$ for $j \in \{0,1\}$ in Line 14. Since $[e^u]_j$ is the share of e^u, the maximum value of the reconstruction process is

$$[e^{u_{max}}]_0 + [e^{u_{max}}]_1 = e^{u_{max}+\check{r}} \cdot 2^{l_f} \cdot ([e^{-\check{r}}]_0 + [e^{-\check{r}}]_1) = e^{u_{max}} \cdot 2^{2l_f}$$

according to Line 14 of Algorithm 2. Therefore, $e^{u_{max}} \cdot 2^{2l_f} \leq 2^{l-1}$ is required to ensure that the reconstruction process does not overflow. As $u < 0$ (i.e., $u_{max} = 0$), it is achieved that PP-Exp does not overflow if $l_f < \frac{l-1}{2}$.

A.2 PROOF OF THEOREM 2

In terms of algorithm security, we analyze the probability of PP-Exp privacy leakage and the *expected degree of information leakage*. The number of possible values of u is m_u and the number of possible values of r is m_r, then the probability that the algorithm leads to additional privacy leakage of input u is

$$\frac{(1+2+\cdots+m_u)\cdot 2}{m_u m_r} = \frac{m_u \cdot (m_u - 1)}{m_u m_r} = \frac{m_u - 1}{m_r}. \tag{1}$$

The probability that the PP-Exp is *secure* is $1 - \frac{m_u-1}{m_r} = \frac{m_r - m_u + 1}{m_r}$.

Since the exponents are all negative in the Gaussian process regression, the number of u values is reduced by half. This allows the probability of input u leakage to be further reduced. For example, when $l_f = 2^{29}$, supposing the input u takes values in the range $[-4, 0]$ and r takes values in the range $[-16, 16)$ (i.e., the number of values of u and r are $2^{31}+1$ and $2^{34}+1$, respectively), the probability that the algorithm to be secure is $\frac{2^{34}-2^{31}}{2^{34}} = \frac{7}{8}$.

The *expected degree of information leakage* is used to describe the amount of privacy leakage of the PP-Exp algorithm. It can be understood by the following simple example. Suppose $u \in \{-2,-1,0\}$, $r \in \{-2,-1,0,1,2\}$ and $d = u + r \in$

$\{-4, -3, -2, -1, 0, 1, 2\}$. The probabilities of d taking different values are as follows:

$$\begin{aligned} \Pr\{d=-4\} = \Pr\{d=2\} &= \tfrac{1}{15}, \\ \Pr\{d=-3\} = \Pr\{d=1\} &= \tfrac{2}{15}, \\ \Pr\{d=-2\} = \Pr\{d=-1\} = \Pr\{d=0\} &= \tfrac{3}{15}. \end{aligned} \tag{2}$$

If $d \in \{-2, -1, 0\}$, u can take any of $-2, -1, 0$ such that no additional information about u is revealed by d. Therefore, the degree of information leakage of u is $\frac{3}{15} \cdot \frac{1}{3} = \frac{1}{5}$. If $d \in \{-4, 2\}$, u will only correspond to unique values, and the degree of information leakage of u is $2 \cdot \frac{1}{15} \cdot 1$. If $d \in \{-3, 1\}$, there are two possible values of u. The degree of information leakage of u is $2 \cdot \frac{2}{15} \cdot \frac{1}{2} = \frac{2}{15}$. Therefore, the *degree of information leakage* of u averaged over $d \in \{-4, -3, 1, 2\}$ is $\frac{2}{15} + \frac{2}{15} + \frac{1}{5} = \frac{7}{15}$.

In PP-Exp, there are $\frac{2k}{m_u m_r}, k = 1, 2, \cdots, m_u - 1$ probabilities that $m_u - k$ possible values of the input u are excluded by d. Therefore, the *average degree of information leakage* of input u is

$$(1 - \frac{m_u - 1}{m_r}) \cdot \frac{1}{m_u} + \sum_{k=1}^{m_u - 1} (\frac{2k}{m_u m_r}) \cdot \frac{1}{k} = \frac{m_r - m_u + 1}{m_u m_r} + \frac{2(m_u - 1)}{m_u m_r} = \frac{m_u + m_r - 1}{m_u \cdot m_r}. \tag{3}$$

It is important to note that the PP-Exp algorithm only exposes the exact value of u when the maximum and minimum values are in the range where u and r are taken simultaneously. This probability is

$$\frac{2}{mn} = \frac{2}{(2^{31}+1) \cdot (2^{34}+1)} < \frac{2}{2^{31} \cdot 2^{34}} = \frac{1}{2^{64}}. \tag{4}$$

B PRIVACY-PRESERVING MATRIX INVERSION

In this section, we will give the construction details of the *privacy-preserving matrix inversion* (PP-MI) operation. The main idea of the PP-MI is to transform the matrix inversion process into MPC-friendly operations such as multiplication and division. Next, we will first briefly recall the process of positive definite matrix inversion via Cholesky decomposition and then provide the pseudo-code of the PP-MI algorithm.

Let $\mathbf{U} = (u_{h,k})_{h,k=1,2,\ldots,n}$ be an $n \times n$ positive definite matrix. It can be decomposed as $\mathbf{U} = \mathbf{L}\mathbf{D}\mathbf{L}^\top$ where $\mathbf{L}$ is a unit lower triangular matrix and $\mathbf{D}$ is a diagonal matrix. Let d_k be the k^{th} diagonal element of $\mathbf{D}$ for $k = 1, \ldots, n$. We can

Algorithm 1 Privacy-preserving Cholesky decomposition

Setup. The servers determine an integer ring $\mathcal{Z}_L$.
Input. S_0 holds the share $[\mathbf{U}]_0$; S_1 hold the share $[\mathbf{U}]_1$.

```
1: // Offline phase:
2: $T$ generates $\mathbf{A}, \mathbf{B} \in Z_L^{n \times n}$ randomly, and calculates $\mathbf{C} = \mathbf{AB}$;
3: $T$ sends $([\mathbf{A}]_j, [\mathbf{B}]_j, [\mathbf{C}]_j)$ to $S_j$.
4: // Online phase:
5: for $j \in \{0, 1\}$ do:
6:     $[d_1]_j \leftarrow [u_{1,1}]_j$;
7:     $S_j$ calculates $[l_{1:n,1}]_j = [u_{1:n,1}]_j / [u_{1,1}]_j$ by calling PP-Div;
8:     for $k \in \{2, 3, \ldots, n\}$ do:
9:         $S_j$ calculates $[d_k]_j = [u_{k,k}]_j - [\sum_{m=1}^{k-1} l_{k,m}^2 d_m]_j$ by calling PP-MM;
10:        $S_j$ calculates $[\hat{l}_{k+1:n,k}]_j = [u_{k+1:n,k}]_j - [\sum_{m=1}^{k-1} l_{k,m} d_m l_{k+1:n,m}]_j$ by calling PP-MM;
11:        $S_j$ calculates $[l_{k+1:n,k}]_j = [\hat{l}_{k+1:n,k}]_j / [d_k]_j$ by calling PP-Div;
12:    end for
13: end for                                                    ▷ $S_j$ gets $[\mathbf{L}]_j, [\mathbf{D}]_j$.
```

Output. S_j output the share $[L]_j, [D]_j$ for $j \in \{0, 1\}$.

Algorithm 2 Privacy-preserving forward

Setup. The servers determine an integer ring $\mathcal{Z}_L$.
Input. S_0 holds the share $[\mathbf{U}]_0, [\mathbf{L}]_0$; S_1 hold the share $[\mathbf{U}]_1, [\mathbf{L}]_1$.

```
 1: // Offline phase:
 2: $T$ generates $\mathbf{A}, \mathbf{B} \in Z_L^{n\times n}$ randomly, and calculates $\mathbf{C} = \mathbf{AB}$;
 3: $T$ sends $([\mathbf{A}]_j, [\mathbf{B}]_j, [\mathbf{C}]_j)$ to $S_j$.
 4: // Online phase:
 5: $v_{1,1} \leftarrow 1$
 6: for $j \in \{0, 1\}$ do
 7:     for $k \in \{2, 3, \ldots, n\}$ do
 8:         $S_j$ calculates $[v_{k,1:k-1}]_j = -[\sum_{m=1}^{k-1} v_{m,1:k-1} l_{k,m}]_j$ by calling PP-MM;
 9:     end for
10: end for                                                    ▷ $S_j$ gets $[\mathbf{V}]_j$.
```

Output. S_j outputs the share $[\mathbf{V}]_j$ for $j \in \{0, 1\}$.

Algorithm 3 Privacy-preserving backward

Setup. The servers determine an integer ring $\mathcal{Z}_L$.
Input. S_0 holds the share $[\mathbf{V}]_0, [\mathbf{L}]_0, [\mathbf{D}]_0$; S_1 hold the share $[\mathbf{V}]_1, [\mathbf{L}]_1, [\mathbf{D}]_1$.

```
1: // Offline phase:
2: $T$ generates $\mathbf{A}, \mathbf{B} \in Z_L^{n\times n}$ randomly, and calculates $\mathbf{C} = \mathbf{AB}$;
3: $T$ sends $([\mathbf{A}]_j, [\mathbf{B}]_j, [\mathbf{C}]_j)$ to $S_j$.
4: // Online phase:
5: $S_j$ calculates $[\mathbf{\Lambda}]_j = [\mathbf{V}^\top]_j [\mathbf{D}^{-1}]_j [\mathbf{V}]_j$ by calling PP-Div and PP-MM.    ▷ $S_j$ gets $[\mathbf{\Lambda}]_j$.
```

Output. S_j outputs the share $[\mathbf{\Lambda}]_j$ for $j \in \{0, 1\}$.

calculate $\mathbf{L}, \mathbf{D}$ through (5). For $k = 1, 2, \ldots, n$ and $h = k + 1, k + 2, \ldots, n$.

$$\begin{cases} d_k = u_{k,k} - \sum_{m=1}^{k-1} l_{k,m}^2 \cdot d_m \ , \\ l_{h,k} = (u_{h,k} - \sum_{m=1}^{k-1} l_{h,m} \cdot l_{k,m} \cdot d_m)/d_k \ . \end{cases} \tag{5}$$

Supposing $\mathbf{U\Lambda} = \mathbf{I}$ for a given $n \times n$ identity matrix $\mathbf{I}$. Next, we will introduce how to compute $\mathbf{\Lambda}$ from $\mathbf{I}, \mathbf{L}$, and $\mathbf{D}$. Then, we can have $\mathbf{U}^{-1} = \mathbf{\Lambda}$.

Let $\mathbf{V} \triangleq \mathbf{DL}^\top\mathbf{\Lambda}$. Then, $\mathbf{U\Lambda} = (\mathbf{LDL}^\top)\mathbf{\Lambda} = \mathbf{LV} = \mathbf{I}$ and $\mathbf{V}$ is a unit lower triangular matrix. For $k = 1, 2, \ldots, n$ and $h = k + 1, k + 2, \ldots, n$, we can calculate each element of $\mathbf{V}$ as:

$$v_{h,k} = -\sum_{m=1}^{h-1} v_{m,k} \cdot l_{h,m}. \tag{6}$$

we can calculate the matrices $\mathbf{\Lambda}$ using (7):

$$\mathbf{\Lambda} = \mathbf{U}^{-1} = ((\mathbf{LDL}^\top))^{-1} = (\mathbf{L}^{-1})^\top\mathbf{D}^{-1}\mathbf{L}^{-1} = \mathbf{V}^\top\mathbf{D}^{-1}\mathbf{V}. \tag{7}$$

Based on equations (5), (6), and (7), we can implement PP-MI by reasonably invoking privacy-preserving multiplication (PP-MM) and privacy-preserving division (PP-Div) in Knott et al. [2021]. Let $\mathbf{U}$ be the an $n \times n$ positive definite matrix, $u_{a;b,k:h} \triangleq (u_{i,j})_{i=a,\ldots,b}^{j=k,\ldots,h}$. The PP-MI algorithm for $\mathbf{U}$ consists of three parts, and their pseudo-codes refer to the Algorithms 1-3. Note that the pseudo-codes are note exactly the same as (5)-(7) since we vectorize some steps for accelerating the computation.

C COMMUNICATION COMPLEXITY ANALYSIS

In this section, we will analyze the communication complexity of PP-MI step by step. We only analyze the communication complexity in the online phase. The process of the offline phase can be done by the server during idle time.

For PP-MI, we will analyze the communication cost for Algorithms 1, 2 and 3, separately. For Algorithm 1, there is 1 vector-wise division in line 7. For each $k = 2, ..., n-1$, there are 4 rounds of vector-wise multiplication for lines 9-10 and 1 vector-wise division for line 11. When $k = n$, there are only 2 rounds of vector-wise multiplication for line 9. We do not need to compute lines 10-11 when $k = n$. There are 17 rounds of communication for each element/vector-wise PP-Div and 1 round of communication for each element/vector-wise PP-MM. Consequently, the total communication round for Algorithm 1 is $17 + (n-2) * (4 + 17) + 2 = 21n - 23$. Since there are $\mathcal{O}(n^3)$ element-wise multiplications and $\mathcal{O}(n^2)$ element-wise divisions included in Algorithm 1, its communication complexity is $\mathcal{O}(n^3 l)$.

For Algorithm 2, it includes 1 vector-wise multiplication for each $k = 2, \dots, n$ and each round of computation (i.e., line 8 for each k) consists of $(k-1)^2$ element-wise multiplications. Therefore, Algorithm 2 incurs a total of $n-1$ rounds of communication and its communication complexity is $\mathcal{O}(n^3 l)$.

For Algorithm 3, it contains 1 matrix division and 1 matrix multiplication with n^2 elements in line 5. Therefore, the total number of communication rounds required to execute Algorithm 3 is $17 + 1 = 18$. The communication complexity is $\mathcal{O}(n^2 l)$.

In summary, the PP-MI algorithm incurs $21n - 23 + n - 1 + 18 = 22n - 6$ rounds of communication and the communication complexity is $\mathcal{O}(n^3 l)$.

D ADDITIONAL EXPERIMENTAL RESULTS (PP-GPR VS. DP-GPR)

Another technique that has been widely used to achieve privacy preservation in machine learning is *differential-privacy* (DP) [Dwork et al., 2014]. By adding noise that satisfies the privacy budget, DP-based privacy-preserving machine learning algorithms achieve some privacy protection at the expense of the accuracy of the model. In the domain of Gaussian process regression (GPR) algorithms, Smith et al. [2018] proposed a significant contribution by presenting the first privacy-preserving GPR algorithm. However, it is important to note that their approach only safeguards the privacy of the model outputs $\mathbf{y}$ through the addition of noise levels that adhere to a given privacy budget. Consequently, their algorithm falls short of achieving complete protection for both the model inputs and outputs.

To demonstrate the effectiveness of our algorithm, we present the loss of predictive mean of DP-GPR on the diabetes dataset for different levels of DP guarantee (i.e., varying ϵ) using the SE kernel. As can be seen, even with a large privacy budget (i.e., $\epsilon = 1.0$), DP-GPR incurs significantly larger computational errors due to the additional DP noise compared to the proposed PP-GPR. The $Loss_\mu$ of our proposed algorithm is kept below 0.01% (Table 2 in the main paper).

Table 1: Comparison of our algorithm with the DP-based algorithm in the relative difference $Loss_\mu$.

n	Test	$\epsilon = 1.0$	$\epsilon = 0.5$	$\epsilon = 0.2$	**Ours**
80	20	$18.6\%(\pm 23.4)$	$42.9\%(\pm 188.3)$	$96.6\%(\pm 584.8)$	$0.0007\%(\pm 3.6e-04)$
150	50	$27.8\%(\pm 19.4)$	$61.2\%(\pm 311.6)$	$148.6\%(\pm 1457.7)$	$0.0018\%(\pm 1.3e-03)$
300	100	$18.3\%(\pm 10.5)$	$33.5\%(\pm 23.7)$	$80.6\%(\pm 133.3)$	$0.0058\%(\pm 2.5e-03)$

References

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

Brian Knott, Shobha Venkataraman, Awni Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. CrypTen: Secure multi-party computation meets machine learning. In *Proc. NeurIPS*, 2021.

Michael T Smith, Mauricio A Álvarez, Max Zwiessele, and Neil D Lawrence. Differentially private regression with Gaussian processes. In *Proc. AISTATS*, pages 1195–1203, 2018.