Efficient, Secure, Differentially Private Deep Learning in the Two-Server Model

Jun Feng[1], Hong Sun[2], Pengfei Zhang*[3], Bocheng Ren[4], Shunli Zhang[5]
[1]Hubei Key Laboratory of Distributed System Security, Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology
[2]School of Economics, Wuhan Textile University
[3]School of Computer Science and Engineering, Anhui University of Science and Technology, and Key Laboratory of Equipment Data Security and Guarantee Technology, Ministry of Education, Guilin University of Electronic Technology
[4]School of Computer Science and Technology, Hainan University
[5]School of Computer and Information Science, Qinghai Institute of Technology
junfeng@hust.edu.cn, hsun@wtu.edu.cn, zpf.bupt@bupt.cn, bc.revincent@gmail.com, shunlizh@qhit.edu.cn

Abstract

Existing solutions on differentially private deep learning (DPDL) either require the assumption of a trusted data server (centralized DPDL) or suffer from poor utility (local DPDL); and hence their adoptions are hampered in real-world scenarios.We present CRYPTDP, a crypto-assisted differentially private deep learning approach in the two-server model. CRYPTDP employs two non-colluding servers to collaboratively and efficiently train differentially private deep learning over the secret shares of data owners' private data while protecting the confidentiality of the data from untrusted servers. CRYPTDP is the first approach with the best of both local DPDL and centralized DPDL models, which does not resort to trusted server like local DPDL and has the utility like centralized DPDL. In particular, we also make innovations for addressing the major challenges like poor performance and security that beset CRYPTDP: We introduce a new secure computation and differential privacy friendly activation function; we propose a novel garbled-circuits-free most significant bit extraction protocol, and using the protocol we propose an efficient and secure garbled-circuits-free protocol for activation function over secret shares. Exhaustive experiments show that CRYPTDP delivers significantly better performance than the state-of-the-art local DPDL, yields higher accuracy than the state-of-the-art centralized DPDL, and can achieve two orders of magnitude faster runtime than the state-of-the-art approach.

Introduction

More recently, two general approaches have been proposed that consider differentially private deep learning (DPDL) to protect the private data against the adversaries: Centralized differentially private deep learning (CDPDL) (Abadi et al. 2016; Papernot et al. 2020; Wang et al. 2025) and local differentially private deep learning (LDPDL) (Arachchige et al. 2019; Lyu et al. 2020). The former can provide accuracy guarantees for released results, but rely on the assumption of a trusted server, which has full access to all training data in clear. The latter requires no trusted server, but is subjected to poor utility even for more samples. Unfortunately, none of these approaches can implement the DPDL without the requirement of a trusted server (like LDPDL) and with better utility (like CDPDL) (Feng et al. 2025). The absence of such an approach renders the adoption of DPDL impractical in the real world; however, implementing the approach is challenging and non-trivial (Zhang et al. 2025a,b). Therefore, in this work, we consider the critical problem: How to design an efficient approach to get the best of both CDPDL and LDPDL worlds for practical DPDL.

To tackle the above problem, we employ the two-server model (Böhler and Kerschbaum 2020b; Miao et al. 2024; Balle et al. 2025) for differentially private deep learning. In this model, two servers run a differentially private deep learning approach to train a model without revealing any information on data owners' data (Böhler and Kerschbaum 2020a; Feng et al. 2020). The approach is obtained via our deep-learning/differential-privacy/secure-computation codesign. The approach relies on non-linear functions, which make the problem even more challenging (Ren et al. 2025a,b). 1) In prior works, no activation function is suitable for both secure deep learning and differentially private deep learning. To increase the efficiency of secure activation, prior works make use of square function in place of ReLU for secure deep learning. However, the method degrades the accuracy of secure deep learning. CDPDL needs gradient clipping and adding noise, both ReLU and square function also affect the accuracy of deep learning with differential privacy. The off-the-shelf activation functions suffer from either inefficiency or accuracy problems when they are used in secure and differentially private deep learning. 2) A large number of non-linear operations are needed for deep learning training. The protocols for nonlinear functions in DELPHI (Mishra et al. 2020) and Cheetah (Huang et al. 2022) are designed for the client-server model, and the protocols in Falcon (Wagh et al. 2021b), SWIFT (Koti et al. 2021), Rabbit (Makri et al. 2021), BLAZE (Patra and Suresh 2020) are designed for the three server model, while our protocols are designed for the two-server model. The three-server and two-server models have different settings. They cannot be used in the two-server model. In practice, it is more difficult to find three servers than two servers. The protocols for nonlinear functions in SecureML(Mohassel

*Corresponding author.

and Zhang 2017), ABY2. 0 (Patra et al. 2021), CRYPTEN (Knott et al. 2021), and SIRNN (Rathee et al. 2021) use GC-based comparison for the two-server model. All of the existing protocols for non-linear functions in the ciphertext domain rely on garbled circuits (Xie et al. 2025; Saleem et al. 2024). While we can use the protocols for nonlinear functions, their performance remains a major issue due to the use of the costly garbled circuits over secret shares in the two-server model. Therefore, it is imperative to take into account a new activation function, new efficient protocols with needing no garbled circuits for non-linear function when we design the approach.

Contributions. In this work, we propose CRYPTDP, a novel crypto-assisted differentially private deep learning, which overcomes the aforementioned challenges. CRYPTDP trains deep learning with differential privacy while it provides high accuracy, requires no trusted server, and has high efficiency. The contributions are described as follows.

New Approach for Differentially Private Deep Learning. We present CRYPTDP for differentially private deep learning on untrusted servers. Through the well codesign of deep learning, differential privacy and secure computation, CRYPTDP enables two non-colluding and untrusted servers to securely and efficiently perform differentially private deep learning training, where data owners' data are $\langle\cdot\rangle$-shared among the two servers. We introduce a new privacy definition called 2PC-zCDP, which combines zCDP and secure 2-party computation (2PC), for the provable privacy guarantees of CRYPTDP. To the best of our knowledge, CRYPTDP, for the first time, achieves the best of both worlds for differentially private deep learning, which has no need for a trusted server like the LDPDL approaches and achieves the accuracy guarantees of the CDPDL approaches. CRYPTDP does not reveal any information about the private data; and the released differentially private model can defend against inference attacks so that an adversary is not able to infer the data during the training. We also show that the OME-based LDPDL model severely underestimates its privacy loss.

New Activation Function and Novel Garbled-Circuits-Free Protocols. CRYPTDP incorporates several novel approaches for optimizing both model accuracy and running time. We introduce a new secure computation and differential privacy friendly activation function tailored to CRYPTDP. The function considers jointly accuracy and efficiency, which enables effective and efficient CRYPTDP. We present a novel garbled-circuits-free most significant bit extraction (MSBExtr) protocol. Leveraging the MSBExtr protocol and oblivious transfer, we carefully devise efficient and secure oblivious protocols that use no garbled circuits and hide access patterns for nonlinear functions including the new activation in private deep learning. To the best of our knowledge, the protocols are first to use cheap multiplications in place of expensive garbled circuits to essentially reduce the cost per nonlinear function over secret shares, which enables significant efficiency improvements over existing protocols. We call these garbled-circuits-free protocols of nonlinear functions. Furthermore, thanks to access pattern hiding, the protocols can defend against the attacks using access pattern leakage.

Evaluations. We show that CRYPTDP satisfies ρ-2PC-zCDP in the semi-honest model. Our CRYPTDP system is implemented and its performance is evaluated. The extensive evaluations demonstrate that CRYPTDP is superior in terms of accuracy and running time on real-world datasets. CRYPTDP provides significantly high accuracy compared to the OUE/SUE-based LDPDL model, even while spending small privacy budgets; CRYPTDP also remains competitive against the OME-based LDPDL model that fails to achieve differential privacy. CRYPTDP obtains high accuracy gains compared to the state-of-the-art CDPDL. Our protocols for the activation and the max pooling are prominently faster than the corresponding state-of-the-art protocols. CRYPTDP can outperform the corresponding state-of-the-art approach by two orders of magnitude in runtime.

Proposed Constructions

In this section, we elaborate a series of core secure protocols over secret-shared values to implement CRYPTDP, and show the security of CRYPTDP. The protocols comprise privacy-preserving activation (PPActi), privacy-preserving pooling (PPPool), privacy-preserving gradient approximation (PPGA), and privacy-preserving convolution (PPConv). We minimize the runtime of CRYPTDP without disclosing the privacy of data via our deep-learning/differential-privacy/secure-computation codesign.

Activation Function and Its Protocol

New Activation Function Activation functions are important in deep learning for determining the non-linear mapping relation between the input and the output. While rectified linear unit (ReLU) has been found to perform better and is the recommended activation function for convolutional neural networks (CNNs) in non-private situations, it does not always hold in private situations, especially in deep learning with differential privacy.

In CDPDL, the effect of each example on the gradient is bounded by clipping such that sensitivity is bounded. We find that unbounded activation functions cause larger gradient magnitudes, which result in more information losses when per-example gradient clipping is used for CDPDL. To overcome the problem, we propose a new activation function, which is defined by Equation 1:

$$f(z) = \begin{cases} 0, & \text{if } z \leq 0 \\ z, & \text{if } 0 < z < 1 \\ 1, & \text{if } z \geq 1 \end{cases} . \tag{1}$$

The function is both secure computation and differential privacy friendly. The function replaces all negative values in the feature map by zero and all values greater than 1 by 1. On one hand, the function only involves comparison operations; it therefore can be efficiently and securely evaluated by using secure computation techniques. On the other hand, the new activation function is bounded. Gradient clipping is required to achieve differential privacy in DPDL; however, it induces information loss. ReLU has no upper bound.

Compared with ReLU, the new activation function can help reduce information loss, which can improve the accuracy of DPDL. The new activation function helps control the magnitude of gradient, which in turn reduces information loss by clipping. This improves the accuracy of DPDL. The effectiveness of the function is also confirmed with experiments in Section . As a result, the function is also effective for DPDL. We note that the new activation function is the first function that allows both efficient secure computation and effective differential privacy for private deep learning. The new function is superior to prior activation functions for CRYPTDP.

Garbled-Circuits-Free Most Significant Bit Extraction Consider two servers C_0 and C_1 with a value a additively shared among them who want to extract the boolean shares of the most significant bit (MSB) of a. It is inefficient to apply garbled circuit protocol to extract MSB over the shares, since we need to convert the arithmetic sharing to Yao sharing, evaluate the function with garbled circuits, and convert back to arithmetic sharing. However, we make the observations: To determine the MSB (or sign) of the value a, we can use sign rule of multiplication. For the multiplication $c = a \cdot b$, if server C_0 has the MSB $bMSB$ of b, and server C_1 can get the MSB $cMSB$ of c, then the sign $aMSB$ of a can be determined by $bMSB$ and $cMSB$, so the problem becomes how to design a protocol such that the MSB of multiplicand is known only to server C_0, the MSB of result is known only to server C_1, multiplier and its MSB are not known to servers C_0 and C_1; The multiplication over the secret shares can be computed cheaply. Let $[\lambda_c]_0 + [\lambda_c]_1 = \lambda_a\lambda_b$. The multiplication can be written:

$$\begin{aligned} c = a \cdot b &= (\Delta_a - \lambda_a) \cdot (\Delta_b - \lambda_b) \\ &= \Delta_a\Delta_b - \lambda_a\Delta_b - \lambda_b\Delta_a + \lambda_a\lambda_b \\ &= (\Delta_a\Delta_b - \lambda_b\Delta_a - [\lambda_a]_0\Delta_b + [\lambda_c]_0) \\ &+([\lambda_c]_1 - [\lambda_a]_1\Delta_b). \end{aligned} \tag{2}$$

Building on the above key observations, we propose a novel MSB Extraction (MSBExtr) protocol for the $\langle\cdot\rangle$-sharing, which uses multiplication instead of expensive garbled circuits. Our protocol consists of an offline phase and an online phase, and shifts some operations to the offline phase. Our approach enables a very efficient online phase for protocols employing the proposed MSBExtr protocol.

The MSBExtr protocols considers server C_0 with secret share $\langle a\rangle_0 = (\Delta_a, [\lambda_a]_0)$ and server C_1 with secret share $\langle a\rangle_1 = (\Delta_a, [\lambda_a]_1)$, where $a = \Delta_a - [\lambda_a]_0 - [\lambda_a]_1$, and returns the binary secret share $[aMSB]_i$ of the MSB $aMSB$ of a as output to server C_i for $i \in \{0, 1\}$ without revealing a and $aMSB$ to server C_0 and C_1, where $aMSB = [aMSB]_0^B \oplus [aMSB]_1^B$. The overall steps of the MSBExtr protocol are described formally in Algorithm 1. We formally define the MSBExtr protocol as follows:

$$([aMSB]_0^B, [aMSB]_1^B) \leftarrow MSBExtr(\langle a\rangle). \tag{3}$$

In what follows, we elaborate the MSBExtr protocol. During the offline (setup) phase, the multiplication triplets $([\lambda_a], \lambda_b, \lambda_c)$ are generated by executing $setupMULT([\lambda_a], \lambda_b)$ such that server C_0 has $([\lambda_c]_0, \lambda_b)$ and server C_1 has share $[\lambda_c]_1$, where $[\lambda_c]_0 + [\lambda_c]_1 = \lambda_a\lambda_b$; server C_0 gets randoms $b, \Delta_b \in Z_{2^k}$, where $\Delta_b = b + \lambda_b$, and sets $[aMSB]_0^B$ to be the most significant bit (MSB) of b, and server C_1 gets Δ_b. Note that b is only held by server C_0, hence server C_1 does not know $[aMSB]_0^B$. The offline phase is input independent, and can be performed before the actual inputs are known. The setup phase can be instantiated with either oblivious transfer or homomorphic encryption by two servers or can be generated by clients.

Algorithm 1: Most Significant Bit Extraction (MSBExtr) Protocol on the $\langle\cdot\rangle$-Shares

Input: The $\langle\cdot\rangle$-sharing $\langle a\rangle$ of a among servers C_0 and C_1.
Output: Server C_i gets share $[aMSB]_i$ of the MSB of a for $i \in \{0, 1\}$.

1: **Setup:**
2: Server C_0 gets $([\lambda_c]_0, \lambda_b)$ and server C_1 gets $[\lambda_c]_1$ where $[\lambda_c]_0 + [\lambda_c]_1 = \lambda_a\lambda_b$.
3: Server C_0 gets randoms $b, \Delta_b \in Z_{2^k}$ where $\Delta_b = b+\lambda_b$ and server C_1 gets Δ_b. Server C_0 sets $[aMSB]_0^B$ to be the MSB of b.
4: **Online:**
5: Server C_0 locally computes and sends to server C_1: $[c]_0 = \Delta_a\Delta_b - \lambda_b\Delta_a - [\lambda_a]_0\Delta_b + [\lambda_c]_0$.
6: Server C_1 locally computes $[c]_1 = [\lambda_c]_1 - [\lambda_a]_1\Delta_b$.
7: Server C_1 locally computes $c = [c]_0 + [c]_1$ and sets $[aMSB]_1^B$ to be the MSB of c.

In the online phase, taking advantage of the multiplication triplets, servers C_0 and C_1 jointly and securely calculate shares $[c]_0$ and $[c]_1$ such that $c = a \cdot b$ (steps 5-6). First of all, server C_0 locally calculates the value $[c]_0 = \Delta_a\Delta_b - \lambda_b\Delta_a - [\lambda_a]_0\Delta_b + [\lambda_c]_0$ and sends $[c]_0$ to server C_1 (step 5). Server C_1 locally calculates the value $[c]_1 = [\lambda_c]_1 - [\lambda_a]_1\Delta_b$ (step 6). Finally, after receiving $[c]_0$, server C_1 locally calculates the value $c = [c]_0 + [c]_1$ and sets $[aMSB]_1^B$ to be the MSB of c (step 7).

In the MSBExtr protocol, the value a is masked by using $[\lambda_a]_0$ and $[\lambda_a]_1$. The value λ_a is not available in clear to any of two servers C_0 and C_1. The value b is masked by using λ_b which is not known by server C_1, thus server C_1 does not know b. The value c is obtained in clear by server C_1 in step 7, which equals $a \cdot b$. The share $[c]_1$ is known only to server C_1. Only server C_1 knows c and $[aMSB]_1^B$. Therefore, server C_0 and C_1 cannot calculate the value a. Note that for communication, the online of the MSBExtr protocol only involves one round with sending just one element.

Similar to 1-input MSBExtr protocol, given the $\langle\cdot\rangle$-sharing $\langle a\rangle$ and $\langle d\rangle$ of a and d, we can design the MSBExtr2 protocol without garbled-circuits to get the binary secret share $[adMSB]_i$ of the MSB $adMSB$ of $a \cdot d$ over $\langle a\rangle$ and $\langle b\rangle$:

$$([abMSB]_0^B, [abMSB]_1^B) \leftarrow MSBExtr2(\langle a\rangle, \langle b\rangle). \tag{4}$$

The online communication of the MSBExtr2 protocol also only involves sending one element.

Garbled-Circuits-Free Activation Protocol Prior works almost exclusively focus on the designs and optimiza-

tions (such as SecureML (Mohassel and Zhang 2017) and ABY2.0 (Patra et al. 2021)) of the garbled circuits based protocols for privacy-preserving activation, but the protocols are still inefficient because of the expensive garbled circuits. In addition, as seen from Equation 1, the activation function makes use of input-dependent conditional branches, which result in input-dependent access patterns. Information may be disclosed from the access patterns. However, this remains an unsolved problem in existing protocols.

We re-think existing protocols and carefully design a novel more involved oblivious garbled-circuits-free PPActi protocol. The main goal of the protocol is that servers C_0 and C_1 jointly and securely evaluate the activation function along with its derivative. Using the proposed MSBExtr protocol and OT, the protocol radically improves efficiency by not using expensive garbled circuits, and hides access patterns. In the design of the protocol, we minimize the rounds of interaction and number of invoked OTs. Let $\delta_{f(z)}$ represent the derivative of the new activation function $f(z)$. $\delta_{f(z)}$ can be expressed as: $\delta_{f(z)} = 1$ if $0 < z < 1$; and $\delta_{f(z)} = 0$ otherwise. This protocol considers server C_0 with secret share $\langle z \rangle_0 = (\Delta_z, [\lambda_z]_0)$ and server C_1 with secret share $\langle z \rangle_1 = (\Delta_z, [\lambda_z]_1)$, and returns the additive secret share $\langle f(z) \rangle_i$ of the activation result and the additive secret share $\langle \delta_{f(z)} \rangle_i$ of its derivative result as outputs to server C_i for $i \in \{0, 1\}$. For correctness to hold, the following relations will be needed: $f(z) = \Delta_{f(z)} - [\lambda_{f(z)}]_0 - [\lambda_{f(z)}]_1$ and $\delta_{f(z)} = \Delta_{\delta_{f(z)}} - [\lambda_{\delta_{f(z)}}]_0 - [\lambda_{\delta_{f(z)}}]_1$. Both the activation and its derivative can be securely computed together in one iteration. The derivative result shares $\langle \delta_{f(z)} \rangle_0$ and $\langle \delta_{f(z)} \rangle_1$ are employed in back-propagation computation.

We set $I = 1 \times 2^l$, where l is the bit-length of the fractional part in data representation. Let s_1 and s_2 represent the MSBs of z and $z-1$, respectively. We observe that Equation 1 can be rewritten as $f(z) = ((!s_1) \wedge s_2)z + (!s_2)$, and that $\delta_{f(z)}$ can also be expressed as $\delta_{f(z)} = (!s_1) \wedge s_2$. Here $(!s_1) \wedge s_2$ is equal to the MSB s_{12} of $z \cdot (z-1)$. The MSBs can be got by using the proposed MSBExtr protocol. Building on the above key observations, the PPActi protocol is proposed. Servers C_0 and C_1 first work collaboratively to determine the Boolean shares of the MSBs s_2 and s_{12} by using the MSBExtr protocol from Section . Next, servers C_0 and C_1 perform secure forward pass and secure backward pass to compute $((!s_1) \wedge s_2)z + (!s_2)$ and $(!s_1) \wedge s_2$ for our activation by using OT. During this protocol, only server C_0 knows $[s_2]_0^B$ and $[s_{12}]_0^B$, while only server C_1 knows $[s_{12}]_1^B$ and $[s_2]_1^B$. No information regarding z is revealed to servers C_0 and C_1. Neither server C_0 nor server C_1 knows s_1, s_2 and s_{12}. The overall steps of the PPActi protocol are described formally in Algorithm 2. In what follows, we elaborate the protocol.

MSB Extraction. First of all, servers C_0 and C_1 execute the proposed MSBExtr protocol, $([s_2]_0^B, [s_2]_1^B) \leftarrow MSBExtr(\langle zI \rangle)$ (steps 1-2), on $\langle zI \rangle$ such that server C_0 gets the share $[s_2]_0^B$ of the MSB s_2 of $z-1$ and server C_1 gets the share $[s_2]_1^B$. Then, servers C_0 and C_1 call the proposed MSBExtr2 protocol, $([s_{12}]_0^B, [s_{12}]_1^B) \leftarrow MSBExtr2(\langle z \rangle, \langle zI \rangle)$ (step 3), on $\langle z \rangle$ and $\langle zI \rangle$ such that server C_0 gets the share $[s_{12}]_0^B$ of the MSB s_{12} of $z \cdot (z-1)$ and server C_1 gets the share $[s_{12}]_1^B$. Note that only server C_0 knows $[s_2]_0^B$ and $[s_{12}]_0^B$, and only server C_1 knows $[s_2]_1^B$ and $[s_{12}]_1^B$.

Algorithm 2: PPActi Protocol on the $\langle \cdot \rangle$-Shares

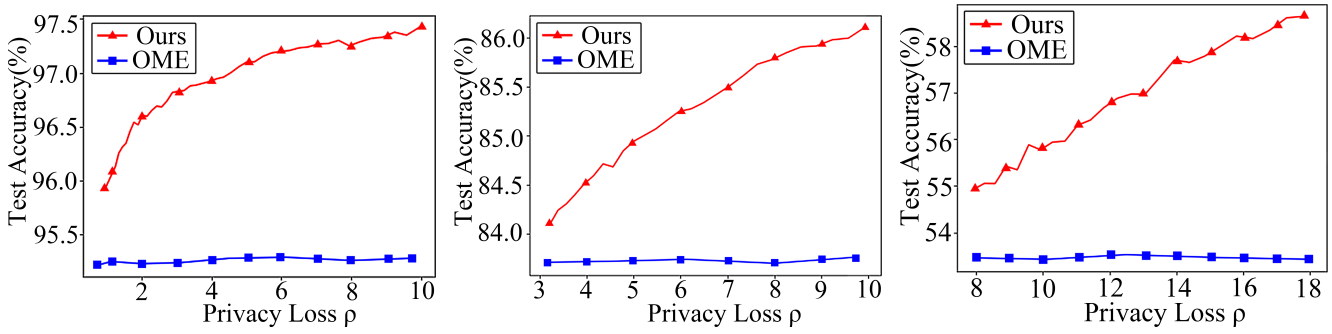
Input: Server C_i ($i \in \{0, 1\}$) holds share $\langle z \rangle_i$ of z.
Output: Server C_i ($i \in \{0, 1\}$) gets shares $\langle f(z) \rangle_i$ and $\langle \delta_{f(z)} \rangle_i$.

1: Server C_i locally computes $\langle zI \rangle = \langle z \rangle - I$ for $i \in \{0, 1\}$.
2: Servers C_0 and C_1 invoke the MSBExtr protocol on $\langle zI \rangle$ to learn output $[s_2]^B$.
3: Servers C_0 and C_1 invoke the MSBExtr2 protocol on $< z >$ and $< zI >$ to learn output $[s_{12}]^B$.
4: C_0 sets $t_{1\_0} = \langle z \rangle_0 \cdot [s_{12}]_0^B + r_1$ and $t_{1\_1} = \langle z \rangle_0 \cdot (![s_{12}]_0^B) + r_1$. C_0 and C_1 perform $(\perp, t_1) \leftarrow OT(t_{1\_0}, t_{1\_1}, [s_{12}]_1^B)$.
5: C_1 sets $t_{2\_0} = \langle z \rangle_1 \cdot [s_{12}]_1^B + r_2$ and $t_{2\_1} = \langle z \rangle_1 \cdot (![s_{12}]_1^B) + r_2$. C_0 and C_1 perform $(\perp, t_2) \leftarrow OT(t_{2\_0}, t_{2\_1}, [s_{12}]_0^B)$.
6: C_0 sets $t_{3\_0} = (![s_2]_0^B) + r_3$ and $t_{3\_1} = [s_2]_0^B + r_3$. C_0 and C_1 perform $(\perp, t_3) \leftarrow OT(t_{3\_0}, t_{3\_1}, [s_2]_1^B)$.
7: Server C_0 locally sets $\langle f(z) \rangle_0 = t_2 - r_1 - r_3$, and server C_1 locally sets $\langle f(z) \rangle_1 = t_1 + t_3 - r_2$.
8: C_0 sets $t_{4\_0} = [s_{12}]_0^B + r_4$ and $t_{4\_1} = (![s_{12}]_0^B) + r_4$. C_0 and C_1 perform $(\perp, t_4) \leftarrow OT(t_{4\_0}, t_{4\_1}, [s_{12}]_1^B)$.
9: Server C_1 gets a random $r_4{}' \in Z_{2^k}$, locally computes and sends to server C_0: $\Delta_{\delta_{f(z)}} = t_4 + r_4{}'$.
10: Server C_0 locally sets $\langle \delta_{f(z)} \rangle_0 = (\Delta_{\delta_{f(z)}}, r_4)$ and server C_1 locally sets $\langle \delta_{f(z)} \rangle_1 = (\Delta_{\delta_{f(z)}}, r_4{}')$.

Forward Pass for Our Activation. Servers securely calculate the $\langle \cdot \rangle$-sharing of our activation. Firstly, server C_0 picks a random number r_1 and sets t_{1_0} and t_{1_1} as follows: $t_{1\_0} = \langle z \rangle_0 \cdot [s_{12}]_0^B + r_1$ and $t_{1\_1} = \langle z \rangle_0 \cdot (![s_{12}]_0^B) + r_1$. Server C_0 with inputs t_{1_0} and t_{1_1} and server C_1 with an input choice bit $[s_{12}]_1^B$ invoke the OT protocol $(\perp, t_1) \leftarrow OT(t_{1\_0}, t_{1\_1}, [s_{12}]_1^B)$ where server C_0 is the sender and server C_0 is the receiver (step 4). After OT invocation, server C_1 obtains t_1, which is equivalent to $\langle z \rangle_0 \cdot ([s_{12}]_0^B \oplus [s_{12}]_1^B) + r_1$ or $\langle z \rangle_0 \cdot (s_1 \oplus s_2) + r_1$.

Then, server C_1 picks a random number r_2 and sets $t_{2\_0} = \langle z \rangle_1 \cdot [s_{12}]_1^B + r_2$ and $t_{2\_1} = \langle z \rangle_1 \cdot (![s_{12}]_1^B) + r_2$. Server C_0 having a choice bit $[s_{12}]_0^B$ and server C_1 having messages t_{2_0} and t_{2_1} perform the OT protocol $(\perp, t_2) \leftarrow OT(t_{2\_0}, t_{2\_1}, [s_{12}]_0^B)$ (step 5). Server C_0 receives t_2 as its output, which is equal to $\langle z \rangle_1 \cdot ([s_{12}]_0^B \oplus [s_{12}]_1^B) + r_2$ or $\langle z \rangle_1 \cdot (s_1 \oplus s_2) + r_2$.

After this, server C_0 picks a random number r_3 and sets $t_{3\_0} = (![s_2]_0^B) + r_3$ and $t_{3\_1} = [s_2]_0^B + r_3$. Servers engage in the OT protocol $(\perp, t_3) \leftarrow OT(t_{3\_0}, t_{3\_1}, [s_2]_1^B)$ with server C_0 being the sender and server C_1 being the receiver (step 6). Server C_1 learns t_3 as its output, which is equal to $([s_2]_0^B \oplus [s_2]_1^B) + r_3 = (!s_2) + r_3$.



(a) MNIST (b) FashionMNIST (c) CIFAR10

Figure 1: Accuracy comparison of our CRYPTDP model with the OME-based LDPDL model on MNIST, FashionMNIST and CIFAR10.

At the end, server C_0 obtains the output share $\langle f(z)\rangle_0$ of activation $f(z)$ by locally setting $\langle f(z)\rangle_0 = t_2 - r_1 - r_3$, while server C_1 obtains the output share $\langle f(z)\rangle_1$ by locally setting $\langle f(z)\rangle_1 = t_1 + t_3 - r_2$ (step 7). $(\langle f(z)\rangle_0, \langle f(z)\rangle_1)$ forms a $\langle\cdot\rangle$ sharing of $f(z)$. Here $\Delta_{f(z)} - [\lambda_{f(z)}]_0 - [\lambda_{f(z)}]_1$ is equal to $f(z)$.

Backward Pass for Our Activation. The rest of this protocol can be used to securely calculate the $\langle\cdot\rangle$-sharing of the derivative of our activation. Server C_0 picks a random number r_4 and sets $t_{4\_0} = [s_{12}]_0^B + r_4$ and $t_{4\_1} = (![s_{12}]_0^B) + r_4$. Server C_0 with messages t_{4_0} and t_{4_1} as the sender and server C_1 with a choice bit $[s_{12}]_1^B$ as the receiver run the OT protocol $(\perp, t_4) \leftarrow OT(t_{4\_0}, t_{4\_1}, [s_{12}]_1^B)$ (step 8). Server C_1 obtains t_4 as its output, which is equal to $([s_{12}]_0^B \oplus [s_{12}]_1^B) + r_4 = (s_1 \oplus s_2) + r_4$. Then, server C_1 gets a random $r_4' \in Z_{2^k}$, locally computes $\Delta_{\delta_{f(z)}} = t_4 + r_4'$ and sends $\Delta_{\delta_{f(z)}}$ to server C_0 (step 9). Finally, servers C_0 and C_1 locally set the derivative output shares $\langle\delta_{f(z)}\rangle_0$ and $\langle\delta_{f(z)}\rangle_1$ as follows: $\langle\delta_{f(z)}\rangle_0 = (\Delta_{\delta_{f(z)}}, r_4)$ and $\langle\delta_{f(z)}\rangle_1 = (\Delta_{\delta_{f(z)}}, r_4')$, respectively (step 10). Here $\Delta_{\delta_{f(z)}} - r_4 - r_4'$ is equal to $\delta_{f(z)}$.

In the PPActi protocol, 4 OTs are performed. The first significant difference between our work and prior works is that only multiplications and OTs are required in the PPActi protocol. Using the proposed MSBExtr protocol, the PPActi protocol avoids expensive garbled circuits, which substantially modifies the prior works. As the experimental data display in Section , the proposed protocol will significantly increase the efficiency of privacy-preserving activation. The second significant difference is that the running of the PPActi protocol is irrespective of input values, which makes that the access patterns are input-independent. It is worth noting that the PPActi protocol is resilient against attacks based on access patterns.

Experimental Evaluation

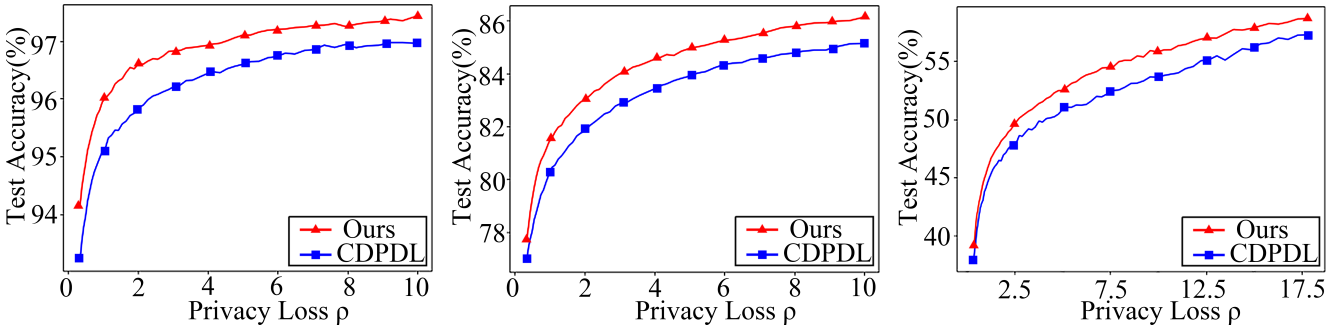
Accuracy Experiments

We employ different convolutional neural network architectures for the MNIST and FashionMNIST datasets, and the CIFAR10 dataset. The choices of architectures are motivated by prior works.

Comparison to Local Differentially Private Deep Learning In this section, we conduct a comparative analysis between the testing accuracy of CRYPTDP and that of the corresponding state-of-the-art local differentially private deep learning (LDPDL) (Arachchige et al. 2019; Lyu et al. 2020). Note that the OME, OUE and SUE based LDPDL methods have been used for textual data in (Lyu et al. 2020) and image data in (Arachchige et al. 2019).

We first compare the CRYPTDP model accuracy with the OME-based LDPDL model (Arachchige et al. 2019; Lyu et al. 2020) for varying privacy loss ρ on the MNIST, FashionMNIST and CIFAR10 datasets in Figure 1. From Figure 1, we observe that the testing accuracy of the CRYPTDP model grows while the testing accuracy of the LDPDL model keeps nearly consistent when ρ increases. For example, Figure 1(a) shows that when ρ increases from 2 to 10, the CRYPTDP testing accuracy grows from 96.52% to 97.42% whereas the LDPDL model keeps nearly 95.25% testing accuracy on MNIST. Another important observation is that CRYPTDP delivers better results than the LDPDL model across various privacy losses ρ. The testing accuracy of CRYPTDP is at least 1.27% higher than that of the LDPDL model. For instance, as shown in Figure 1(a), on MNIST, our model achieves 97.34% testing accuracy for a privacy loss of $\rho = 9$, whereas the LDPDL model only achieves 95.26%. We observe similar results when evaluating on other two datasets FashionMNIST and CIFAR10 as shown in Figures 1(b) and 1(c). Note that a pre-trained deep learning model (whose model architecture is the same as the CRYPTDP model in our experiments) is required to extract the intermediate features before private data perturbation in the OME-based LDPDL model. We remark that this requirement is not reasonable for local differential privacy as a pre-trained deep learning model may not exist in the real world. Furthermore, the OME method fails to protect privacy.

Additionally, we compare the CRYPTDP testing accuracy with the OUE/SUE-based LDPDL models (Arachchige et al. 2019; Lyu et al. 2020). The testing accuracy is summarized in Table 1. As shown in the table, CRYPTDP has significantly higher accuracy than the LDPDL models, even



(a) MNIST (b) FashionMNIST (c) CIFAR10

Figure 2: Accuracy comparison of our CRYPTDP model with the CDPDL model on MNIST, FashionMNIST and CIFAR10.

Dataset	Approach	Accuracy [%]	Privacy Budget
MNIST	OUE	62.22	64
	SUE	65.42	64
	Ours	**97.20**	**3.46**
FashionMNIST	OUE	56.20	64
	SUE	58.45	64
	Ours	**86.11**	**4.47**
CIFAR10	OUE	35.25	64
	SUE	36.06	64
	Ours	**58.63**	**6**

Table 1: Accuracy comparison of our CRYPTDP model with the OUE/SUE-based LDPDL models (Arachchige et al. 2019; Lyu et al. 2020) on MNIST, FashionMNIST and CIFAR10.

though the LDPDL models expend more privacy budgets, on the MNIST, FashionMNIST and CIFAR10 datasets. For example, on FashionMNIST, CRYPTDP reaches a testing accuracy of 86.11% when privacy loss $\epsilon = 4.47$. In contrast, the LDPDL models only have testing accuracies of 56.20% and 58.45% when privacy loss $\epsilon = 64$. The reasons behind this huge difference are as follows: The two LDPDL models are trained on the noisy data and their probabilities fluctuate around 0.5, resulting in low accuracies, whereas CRYPTDP does not suffer from the problems.

In conclusion, the above results demonstrate that CRYPTDP significantly outperforms the current most state-of-the-art LDPDL models.

Comparison to Centralized Differentially Private Deep Learning In this section, we conduct a comparative analysis between the testing accuracy of the CRYPTDP and that of the corresponding state-of-the-art centralized differentially private deep learning (CDPDL) (Yu et al. 2019).

Figure 2 presents the testing accuracy of CRYPTDP and CDPDL under varying privacy loss ρ on the MNIST, FashionMNIST and CIFAR10 datasets. One primary difference between the two models is the activation function used in the hidden layers: CRYPTDP uses the proposed differential-privacy-friendly activation function while the CDPDL model uses ReLU.

The first observation from Figure 2 is that the testing accuracy for both CRYPTDP and CDPDL increases with the increase of the privacy loss ρ. For instance, following from Figure 2(b), when ρ is varied from 2 to 10, the testing accuracy of CRYPTDP increases from 83.02% to 86.11% while the testing accuracy of the CDPDL model increases from 81.83% to 85.14% over the FashionMNIST dataset. A similar trend is observed for the MNIST and CIFAR10 datasets as shown in Figures 2(a) and 2(c).

It is well-known that it is difficult to improve the accuracy of the current CDPDL model. However, another important observation with respect to testing accuracy is that CRYPTDP consistently outperforms the CDPDL model with ReLU across various privacy losses ρ spent. For example, for the MNIST dataset (Figure 2(a)) $\rho = 8$ results in a testing accuracy of 97.25% as compared to testing accuracy of 96.94% for the CDPDL model. Similarly, Figure 2(c) shows that for CIFAR10, CRYPTDP has a testing accuracy of 58.63% for $\rho = 17.5$. In contrast, the CDPDL model has a testing accuracy of 57.25%. The reason behind better accuracy lies in the fact that the proposed differential-privacy-friendly activation function is able to help control the gradient norm.

Our CRYPTDP model mitigates some of the negative impacts from gradient clipping. This conforms with our discussion in Section . It is also worth noting that the CDPDL model only supports a trusted scenario where a trusted data server is required for deep learning model publishing while our CRYPTDP model does not require a trusted data server.

Runtime Experiments

In this section, we report the online time, offline time, and total time of the main building blocks, and analyze how much speed-up is brought about by CRYPTDP.

Activation Experiments In order to evaluate the performance of our activation protocol, we compare our secure activation protocol with the secure activation protocol of ABY2.0 (Patra et al. 2021), which is the current state-of-the-art secure activation protocol for 2-party secure deep learning. The number of samples is fixed to 2000, the number of epochs is fixed to 8, and the mini-batch size is fixed to 16, 64 and 256. If more epochs are required, the running time reported will grow linearly with the number of epochs.

In Table 2, we show the performance of our protocol,

Activation	Method	Batch Size	Total Time
Relu	ABY2.0	16	1013.06
		64	1008.46
		256	999.97
	Ours	16	88.6
		64	78.15
		256	83.04
New Activation	ABY2.0	16	1859.15
		64	1774.09
		256	1754.76
	Ours	16	174.21
		64	153.87
		256	163.76
Softmax	ABY2.0	16	357.32
		64	357.34
		256	354.36
	Ours	16	175.1
		64	176.83
		256	178.07

Table 2: Runtime comparison between our secure activation and the secure activation of ABY2.0 in the whole training. Runtimes are in seconds.

Method	Activation	Softmax	Pooling	GradClip	Total
ABY2.0	678.37	224.01	7037.71	165864.87	173960.9
Ours	40.89	122.03	24.75	454.81	642.78

Table 3: Runtime comparison between CRYPTDP and the approach based on ABY2.0. Runtimes are in seconds.

along with the performance of the protocol using ABY2.0 for all of new activation functions in the whole training. In addition to the new activation function, we also report the time for evaluating secure Relu activation function and Softmax in training.

ABY2.0 employs garbled circuits for secure activation protocol, and we demonstrate that with our protocol, the time taken to evaluate the secure activation protocol is decreased significantly. Our protocol also significantly outperforms in the total time due to the use of multiplication triplets in place of expensive garbled circuits. In particular, we observe that for total time, CRYPTDP is around $10\times$ faster than ABY2.0 in evaluating the new activation while it is around $11\times$ faster in evaluating the Relu activation.

Training Experiments Finally, we compare the running time of CRYPTDP with that of the approach using ABY2.0 (Patra et al. 2021) in Table 3. In the experiments, the number of samples is fixed to 2000, the number of epochs is fixed to 5, and the mini-batch size is fixed to 32. We report the running time of activation, pooling, softmax, and gradient clipping. Following from Table 3, we observe that the running time of CRYPTDP is 642.78s while the running time of the approach based on ABY2.0 (Patra et al. 2021) is 173960.9s. Our CRYPTDP is two orders of magnitude more efficient than the state-of-the-art approach based on ABY2.0. The rationale behind this improvement is the usage of our secure-computation-friendly approaches and our new protocols.

Related Work

Interesting works have been done on the secure computation of differentially private data analysis (Wang et al. 2025; Wagh et al. 2021a; Demelius, Kern, and Trügler 2025). Li et al. (2022) devise new efficient and secure methods based on a trusted execution environment for outsourcing differentially private data publishing. Roy Chowdhury et al. (2020) investigate a new crypto-assisted method, Cryptϵ, to run DP programs for SQL queries in the central model. They employ the two-server model, where one server is one cryptographic server managing the cryptographic primitives while the other is an analytics server carrying out secure computations. However, in our work, both of the servers are analytics servers carrying out secure computations. Böhler and Kerschbaum (2020b,a) present efficient protocols to implement the secure computation of differentially private median of the union of data sets by using the exponential mechanism. The protocols in (Böhler and Kerschbaum 2020b) are carefully designed for the 2-party setting, and engage in the relaxation of DP while the protocols in (Böhler and Kerschbaum 2020a) are designed for the multi-party setting and use pure differential privacy. Böhler and Kerschbaum (2021) also propose new protocols to efficiently and securely evaluate differentially private top-k by using secure multi-party computation. He et al. (2017) propose new private record linkage by composing differential privacy and secure computation. However, their works focus mainly on comparatively simpler data analysis, rather than more involved machine learning algorithms. Kim et al. (2019) present a secure computation approach for differentially private logistic regression. Gil et al. (2020) develop a secure computation for differentially private bayesian learning. Both of the approaches (Kim et al. 2019; Gil et al. 2020) leverage approximate homomorphic encryption resulting in heavy computational costs, while our work leverages secret sharing avoiding the problem. Moreover, the two approaches are designed for distributed setting whereas our approach is for a 2-party setting. Lyu (2020) presents crypto-assisted differentially private collaborative learning using a three-layer encryption approach and distributed differential privacy. This work is also designed for a distributed setting rather than a 2-party setting. Yuan et al. (2021) propose a private deep learning training scheme. A key distinction is that in their work, one computing party holds access to the features, whereas in our scheme, the features remain unknown to all computing parties. These approaches do not study the DPDL problem. Despite these approaches being similar in spirit to CRYPTDP, they are not applicable to the problem considered in our paper.

Conclusion

We present CRYPTDP, a novel approach with the assistance of cryptography for differentially private deep learning in the two-server model. Performance evaluations confirm that CRYPTDP significantly outperforms prior arts.

Acknowledgments

This research was supported by the National Key R&D Program of China (No. 2024YFB3108700), the National Natural Science Foundation of China (No. 62372195, 62472186, and 62462054), CCF-Huawei Populus Grove Fund, the Foundation of Yunnan Key Laboratory of Service Computing (No. YNSC24116), the Key Laboratory of Equipment Data Security and Guarantee Technology, Ministry of Education (No. 2024020300), the Key Laboratory of Computing Power Network and Information Security, Ministry of Education (No. 2024PY010), and the Natural Science Foundation of Wuhan (No. 2024040801020213). The author would like to thank Yongcai Liang and Xuchi Cheng for their help with the experiments and data analysis.

References

Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H. B.; Mironov, I.; Talwar, K.; and Zhang, L. 2016. Deep learning with differential privacy. In *CCS'16*, 308–318.

Arachchige, P. C. M.; Bertok, P.; Khalil, I.; Liu, D.; Camtepe, S.; and Atiquzzaman, M. 2019. Local differential privacy for deep learning. *IEEE Internet of Things Journal*, 7(7): 5827–5842.

Balle, B.; Bell-Clark, J.; Cheu, A.; Gascon, A.; Katz, J.; Raykova, M.; Schoppmann, P.; and Steinke, T. 2025. Hash-Prune-Invert: Improved differentially private heavy-hitter detection in the two-server model. In *IEEE SP'25*, 2903–2918.

Böhler, J.; and Kerschbaum, F. 2020a. Secure multi-party computation of differentially private median. In *USENIX Security'20*, 2147–2164.

Böhler, J.; and Kerschbaum, F. 2020b. Secure sublinear time differentially private median computation. In *NDSS'20*, 1–18.

Böhler, J.; and Kerschbaum, F. 2021. Secure multi-party computation of differentially private heavy hitters. In *CCS'21*, 2361–2377.

Demelius, L.; Kern, R.; and Trügler, A. 2025. Recent advances of differential privacy in centralized deep learning: A systematic survey. *ACM Computing Surveys*, 57(6): 1–28.

Feng, J.; Wu, Y.; Sun, H.; Zhang, S.; and Liu, D. 2025. Panther: Practical secure 2-party neural network inference. *IEEE Trans. Information Forensics and Security*, 20: 1149–1162.

Feng, J.; Yang, L. T.; Zhu, Q.; and Choo, K.-K. R. 2020. Privacy-preserving tensor decomposition over encrypted data in a federated cloud environment. *IEEE Trans. Dependable and Secure Computing*, 17(4): 857–868.

Gil, Y.; Jiang, X.; Kim, M.; and Lee, J. 2020. Secure and differentially private bayesian learning on distributed data. *arXiv preprint arXiv:2005.11007*.

He, X.; Machanavajjhala, A.; Flynn, C.; and Srivastava, D. 2017. Composing differential privacy and secure computation: A case study on scaling private record linkage. In *CCS'17*, 1389–1406.

Huang, Z.; Lu, W.-j.; Hong, C.; and Ding, J. 2022. Cheetah: Lean and fast secure two-party deep neural network inference. In *USENIX Security'22*, 809–826.

Kim, M.; Lee, J.; Ohno-Machado, L.; and Jiang, X. 2019. Secure and differentially private logistic regression for horizontally distributed data. *IEEE Trans. Information Forensics and Security*, 15: 695–710.

Knott, B.; Venkataraman, S.; Hannun, A.; Sengupta, S.; Ibrahim, M.; and van der Maaten, L. 2021. CRYPTEN: Secure multi-party computation meets machine learning. *NeurIPS'21*, 34.

Koti, N.; Pancholi, M.; Patra, A.; and Suresh, A. 2021. SWIFT: Super-fast and robust privacy-preserving machine learning. In *USENIX Security'21*, 2651–2668.

Li, J.; Ye, H.; Li, T.; Wang, W.; Lou, W.; Hou, T.; Liu, J.; and Lu, R. 2022. Efficient and secure outsourcing of differentially private data publishing with multiple evaluators. *IEEE Trans. Dependable and Secure Computing*, 19(1): 67–76.

Lyu, L. 2020. Lightweight crypto-assisted distributed differential privacy for privacy-preserving distributed learning. In *IJCNN'20*, 1–8.

Lyu, L.; Li, Y.; He, X.; and Xiao, T. 2020. Towards differentially private text representations. In *SIGIR'20*, 1813–1816.

Makri, E.; Rotaru, D.; Vercauteren, F.; and Wagh, S. 2021. Rabbit: Efficient comparison for secure multi-party computation. In *FC'21*, 249–270.

Miao, Y.; Yan, X.; Li, X.; Xu, S.; Liu, X.; Li, H.; and Deng, R. H. 2024. RFed: Robustness-enhanced privacy-preserving federated learning against poisoning attack. *IEEE Trans. Information Forensics and Security*, 19: 5814–5827.

Mishra, P.; Lehmkuhl, R.; Srinivasan, A.; Zheng, W.; and Popa, R. A. 2020. Delphi: A cryptographic inference service for neural networks. In *USENIX Security'20*, 2505–2522.

Mohassel, P.; and Zhang, Y. 2017. SecureML: A system for scalable privacy-preserving machine learning. In *SP'17*, 19–38.

Papernot, N.; Thakurta, A.; Song, S.; Chien, S.; and Erlingsson, U. 2020. Tempered sigmoid activations for deep learning with differential privacy. *arXiv preprint arXiv:2007.14191*.

Patra, A.; Schneider, T.; Suresh, A.; and Yalame, H. 2021. ABY2.0: Improved mixed-protocol secure two-party computation. In *USENIX Security'21*.

Patra, A.; and Suresh, A. 2020. BLAZE: Blazing fast privacy-preserving machine learning. *arXiv preprint arXiv:2005.09042*.

Rathee, D.; Rathee, M.; Goli, R. K. K.; Gupta, D.; Sharma, R.; Chandran, N.; and Rastogi, A. 2021. SIRNN: A math library for secure rnn inference. In *IEEE SP'21*, 1003–1020.

Ren, B.; Yang, L. T.; Nie, X.; Feng, J.; Deng, X.; and Zhu, C. 2025a. Zero-shot fault diagnosis for smart process manufacturing via tensor prototype alignment. *IEEE Trans. Neural Networks and Learning Systems*, 36(8): 13983–13994.

Ren, B.; Yi, Y.; Zhang, Q.; and Liu, D. 2025b. Zero-shot image recognition via learning dual prototype accordance across meta-domains. *IEEE Trans. Image Processing*, 34: 6361–6373.

Roy Chowdhury, A.; Wang, C.; He, X.; Machanavajjhala, A.; and Jha, S. 2020. Cryptϵ: Crypto-assisted differential privacy on untrusted servers. In *SIGMOD'20*, 603–619.

Saleem, H.; Ziashahabi, A.; Naveed, M.; and Avestimehr, S. 2024. Hawk: Accurate and fast privacy-preserving machine learning using secure lookup table computation. *arXiv preprint arXiv:2403.17296*.

Wagh, S.; He, X.; Machanavajjhala, A.; and Mittal, P. 2021a. DP-cryptography: Marrying differential privacy and cryptography in emerging applications. *Communications of the ACM*, 64(2): 84–93.

Wagh, S.; Tople, S.; Benhamouda, F.; Kushilevitz, E.; Mittal, P.; and Rabin, T. 2021b. Falcon: Honest-majority maliciously secure framework for private deep learning. *Proceedings on Privacy Enhancing Technologies*, 2021(1): 188–208.

Wang, S.; Dong, C.; Song, X.; Li, J.; Zhou, Z.; Wang, D.; and Wu, H. 2025. Beyond statistical estimation: Differentially private individual computation via shuffling. In *USENIX Security'25*.

Xie, H.; Guo, Y.; Miao, Y.; and Jia, X. 2025. Access-pattern hiding search over encrypted databases by using distributed point functions. *IEEE Trans. Computers*, 74(3): 1066–1078.

Yu, L.; Liu, L.; Pu, C.; Gursoy, M. E.; and Truex, S. 2019. Differentially private model publishing for deep learning. In *SP'19*, 332–349.

Yuan, S.; Shen, M.; Mironov, I.; and Nascimento, A. 2021. Label private deep learning training based on secure multiparty computation and differential privacy. In *NeurIPS 2021 Workshop Privacy in Machine Learning*.

Zhang, P.; Cheng, X.; Zhang, Z.; Zhu, Y.; and Zhang, J. 2025a. Maximizing area coverage in privacy-preserving worker recruitment: A prior knowledge-enhanced geo-indistinguishable approach. *IEEE Trans. Information Forensics and Security*, 20: 5138–5151.

Zhang, P.; Sun, H.; Zhang, Z.; Cheng, X.; Zhu, Y.; and Zhang, J. 2025b. Privacy-preserving recommendations with mixture model-based matrix factorization under local differential privacy. *IEEE Trans. Industrial Informatics*, 21(7): 5451–5459.