FLSwitch: Towards Secure and Fast Model Aggregation for Federated Deep Learning with a Learning State-Aware Switch

Yunlong Mao(✉), Ziqin Dang, Yu Lin, Tianling Zhang, Yuan Zhang,
Jingyu Hua, and Sheng Zhong

State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
maoyl@nju.edu.cn

Abstract. Security and efficiency are two desirable properties of federated learning (FL). To enforce data security for FL participants, homomorphic encryption (HE) is widely adopted. However, existing solutions based on HE treat FL as a general computation task and apply HE protections indiscriminately at each step without considering FL computations' inherent characteristics, leading to unsatisfactory efficiency. In contrast, we find that the convergence process of FL generally consists of two phases, and the differences between these two phases can be exploited to improve the efficiency of secure FL solutions. In this paper, we propose a secure and fast FL solution named FLSwitch by tailoring different security protections for different learning phases. FLSwitch consists of three novel components, a new secure aggregation protocol based on the Pailliar HE and a residue number coding system outperforming the state-of-the-art HE-based solutions, a fast FL aggregation protocol with an extremely light overhead of learning on ciphertexts, and a learning state-aware decision model to switch between two protocols during an FL task. Since exploiting FL characteristics is orthogonal to optimizing HE techniques, FLSwitch can be applied to the existing HE-based FL solutions with cutting-edge optimizations, which could further boost secure FL efficiency.

Keywords: Secure aggregation · Federated learning · Homomorphic encryption · Deep neural network

1 Introduction

Federated learning (FL) [6,35] is a promising paradigm for multiparty collaborative learning. Participants of FL can keep their private training data on devices and send model updates to a central server, which will be responsible for aggregating and updating the model globally. In this way, FL appears to preserve participants' data privacy because no raw data is disclosed explicitly. However,

M. Tibouchi and X. Wang (Eds.): ACNS 2023, LNCS 13905, pp. 476–500, 2023.
https://doi.org/10.1007/978-3-031-33488-7_18

various threats against FL participants have been identified [18,20,36,39,49], including data reconstruction, membership inference, and property inference attacks. To tackle security problems, plenty of studies on secure model aggregation (SMA) have emerged. Briefly, SMA is crucial for secure FL, protecting participants' data privacy from untrusted servers and participants. The existing SMA solutions largely depend on three techniques, i.e., secure multiparty computation (SMC), homomorphic encryption (HE), and differential privacy (DP).

In particular, SMC-based solutions [5,7,46] solve the SMA problem by treating FL as an ordinary multiparty computation protocol and enhancing it with SMC techniques. However, a significant drawback of these solutions is poor scalability. Although great efforts have been made to reduce the overhead for each participant from linear [7] to poly-logarithmic [5] and quadratic [46] in the number of participants, SMC-based large-scale FL is still expensive. HE-based solutions [9,52,55] commonly have good scalability. However, participants' computation and communication costs are huge since FL models have millions of parameters to be encrypted and transmitted. Hence, there is still a gap between the existing HE-based solutions and practical uses. Unlike the previous solutions, DP-based solutions [48,50,57] have no concerns about efficiency because the overhead of perturbing operations is negligible. Nevertheless, it is difficult for DP-based solutions to balance privacy leakage and model usability. Besides, some studies [18,23] have proven that privacy leakage still exists even though a learning process is protected by DP mechanisms.

Since SMA is still an open problem, it is crucial to find an alternative way to meet security and efficiency demands for FL applications. However, we note that achieving an ideal SMA is challenging because several desirable properties should be satisfied by a unified solution: ① Model updates of each SMA participant should be kept confidential to the server and other participants, since private information could be disclosed through model aggregation by various attacks [18,39,49,56]. ② Participants' computation and communication costs should be affordable. An FL task commonly requires incentive computation and heavy communication. If an SMA solution imports expensive operations, the armed FL will become overburdened. ③ It is essential to have good scalability for an SMA solution since FL may serve large-scale users. The overall overhead may be unaffordable if the SMA solution is poor at scalability. ④ An SMA solution should be resistant to participants' dropouts. Otherwise, participants' dropout may cause a failure of SMA solutions.

Unfortunately, both SMC-based and HE-based secure FL solutions have approximated their theoretical efficiency limitations because they treat FL as a standard multiparty protocol while unique characteristics of FL have been ignored. However, we have observed that FL tasks of deep neural networks (DNNs) share a long-tail converging phenomenon even though a fast converging FL scheme is used [29,40]. Through a thorough investigation of the phenomenon, we find that FL tasks commonly have a quick exploring phase where participants negotiate intensively and a slow converging phase when the global model gets relatively stable. Based on this observation, we propose a hybrid SMA solution, FLSwitch, offering fast and secure FL protocols customized to different learning phases. Figure 1 shows the basic idea of FLSwitch, the left side of which indicates the benign FL workflow.

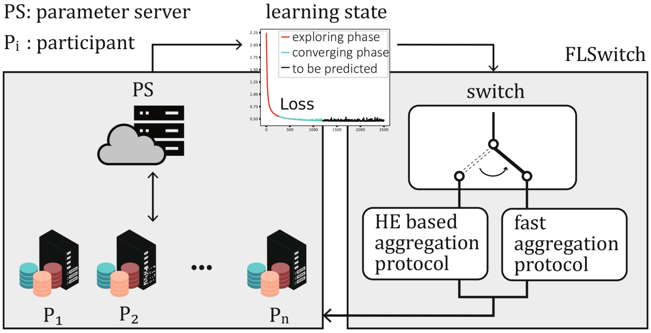


Fig. 1. Workflow illustration of FLSwitch.

Intuitively, FLSwitch consists of two protocols (i.e., HE-based aggregation and fast aggregation protocols indicated in Fig. 1) for different learning states and switches from one to the other when necessary. Although various learning states can be defined in FL tasks, we consider two significantly different states for brevity, referred to as *exploring* and *converging* phases [24,26]. In the exploring phase, FL participants are widely exploring local feature representations. As a result, the total training loss decreases quickly during the exploring phase. Model parameters will also be adjusted intensively. In this case, we design a HE-based protocol, achieving better efficiency than the existing solutions by proposing a residual encoding HE encryption scheme for SMA. In the converging phase, FL participants adjust local models slightly, and the global model state gets relatively stable. To fully utilize the converging phase, we design a fast SMA protocol using a handful of cryptographic operations, further reducing the overhead of learning on ciphertexts.

That leaves a question of determining learning states and switching between protocols. To tackle this problem, we design a state-aware switch model based on meta-learning [13]. During an FL task, the switch keeps watching learning metrics and decides which protocol should be enabled next. Since FL tasks may be divergent and indeterminate, the switch is bidirectional, which means FLSwitch can switch from a HE-based SMA protocol to a fast SMA protocol and vice versa. By integrating all these parts, we get FLSwitch. Please note that utilizing FL characteristics is orthogonal to SMA and other FL studies like participant selection. Hence, this idea can be widely adopted in FL studies. In summary, our contributions are three-fold.

- From the perspective of HE-based SMA, we propose a residual encoding-based HE protocol, outperforming the existing solutions in single instruction

multiple data operating (SIMD), which is verified through analysis and experimental evaluation.
- We propose a fast SMA protocol by utilizing FL characteristics and lightweight cryptographical tools for further efficiency improvement, which significantly speeds up conventional SMA designs.
- To fully utilize the fast aggregation while ensuring FL convergence, we design a switch model based on meta-learning, monitoring FL tasks and switching between protocols dynamically.

2 Preliminary

2.1 Federated Learning

In FL [35,44], a parameter server (PS) coordinates N participants in the same FL task. Each participant $P_i, i \in [1, N]$ has a private dataset for training. Generally, a mini-batch stochastic gradient descent (SGD) optimizer is used by P_i to minimize the loss $\mathcal{L}(\theta^i)$ for local model parameters θ^i. In each iteration, P_i randomly samples training data to construct an input batch $\{x_1, x_2, \ldots, x_B\}$ with the batch size B. Then P_i computes an averaging loss across the batch as $\frac{1}{B}\sum_{j=1}^{B}\mathcal{L}(\theta^i, x_j)$. For updating, the gradient $\mathbf{g}^i$ could be estimated as

$$\boldsymbol{g}^i(\boldsymbol{\theta}^i) = \frac{1}{B}\sum\nolimits_{j=1}^{B}\nabla_{\boldsymbol{\theta}^i}\mathcal{L}(\boldsymbol{\theta}^i, x_j).$$

For the coordination of participants in an FL task, a globally shared training iteration counter $t \in [1, T]$ should be maintained by the PS, assuming that T is an empirically defined maximum iteration number. Denoted by $\boldsymbol{g}_t^i$ the local gradients of P_i in the t-th training iteration. P_i's model parameter $\boldsymbol{\theta}^i$ for the next iteration should be updated by $\boldsymbol{\theta}_{t+1}^i = \boldsymbol{\theta}_t^i - \eta\boldsymbol{g}_t^i$, where η is a predefined hyperparameter for learning rate. After training the model locally, participants (all or selected as indicated in [44]) should upload their updated parameters or gradients to the PS. Then model aggregation will be initiated by the PS. Generally, the PS will perform the model aggregation with a predefined strategy like averaging. In this way, the PS gives the global model

$$\bar{\boldsymbol{\theta}}_{t+1} = \frac{1}{N}\sum\nolimits_{i=1}^{N}\boldsymbol{\theta}_{t+1}^i.$$

At the beginning of the $(t+1)$-th training iteration, all participants should download the latest global model $\bar{\boldsymbol{\theta}}_{t+1}$ from the PS. After synchronizing with the PS, the above steps should be repeated until the global model has achieved the expected performance or the maximum iteration number. We will use $\boldsymbol{\theta}^i$ and $\boldsymbol{\theta}_t^i$ to indicate P_i's local parameters and parameters' state in the t-th iteration. And we will use $\theta_{j,t}^i$ to indicate a specific parameter in the j-th position when $\boldsymbol{\theta}_t^i$ is flattened into a vector, assuming the total amount of parameters is $M, j \in [1, M]$. The superscript and subscript may be omitted if there is no ambiguity.

2.2 Threat Model

Following previous studies [3,5,55], we design FLSwitch in a semi-honest setting, assuming that both the PS and participants could be honest but curious. A secure communication channel is assumed to be available between the PS and each participant. A random secret seed can be pre-installed by a certificate authority or running a distributed protocol between participants only once for secret keys generation during the training. If the PS is adversarial, then no collusion with any adversarial participant is allowed in the semi-honest setting. But we allow the collusion of up to $N-2$ adversarial participants when PS is semi-honest. We will focus on the model confidentiality of participants in the discussion about adversaries, just like in previous studies [5,55]. The correctness and verification of learning will not be discussed here and should be studied separately [14,52].

Adversarial goal. The adversary is to disclose the private information of a target participant. There are plenty of potential attacks against FL participants, such as membership inference [39,45], property inference [30,41], and data reconstruction [18,41] attacks. Some of these attacks should be handled by a mixture of HE and DP techniques. However, DP based solution is orthogonal to our study and should be discussed separately. Hence, we simplify the adversarial goal as disclosing a target participant's local model updates, precisely, model parameters $\boldsymbol{\theta}$ or the corresponding gradients $\boldsymbol{g}$.

2.3 Homomorphic Encryption

We note that our HE-based SMA is implemented on the basis of the original Paillier HE (PHE). But it can also be adapted to other HE systems. For simplicity, we initialize a general PHE system as follows.

- $PSetup(\lambda) \rightarrow (pk, sk)$: Given a security parameter λ, the algorithm generates a pair of public and secret keys (pk, sk).
- $PEncrypt(pk, v) \rightarrow c$: Taking as input a value v and a public key pk, the algorithm outputs a ciphertext c.
- $PDecrypt(sk, c) \rightarrow m$: Taking as input a ciphertext c and a secret key sk, the algorithm outputs the decrypted value v.
- $PAdd(c_1, c_2) \rightarrow c'$: Taking as input two ciphertexts c_1, c_2, the algorithm outputs a ciphertext satisfying $PDecrypt(c_1, sk) + PDecry\ pt(c_2, sk) = PDecrypt(c', sk)$.

3 Secure and Fast Model Aggregation

3.1 FLSwitch Overview

The design rationale of FLSwitch is to handle the SMA problem flexibly with customized protocols for different learning states. Previous studies on SMA barely consider FL characteristics and use a fixed solution for different learning phases.

On the contrary, we exploit the characteristics of different learning phases and design FLSwitch to be aware of the learning state. In this way, we can significantly improve the efficiency and scalability of SMA solutions.

Specifically, we develop a new HE-based SMA protocol for the exploring phase, enabling a more efficient batching method for SIMD operations. Please note that some advancing techniques for adopting HE into FL have been proposed [43,55]. However, different batching methods and ways to solve overflow and quantization problems will result in quite different solutions. Our HE-based solution proposes a novel batching method and new ways to handle overflow and quantization problems, outperforming state-of-the-art HE-based SMA solutions.

Meanwhile, we propose a fast SMA protocol for the converging phase, achieving nearly bare FL efficiency. The basic idea of our fast SMA protocol is to split full-precision model parameters into a stable part and a residual part. In this way, we can find participants holding parameters with the same stable part and encrypt the value only once. If we carefully choose the precision of the stable part, we can always obtain a set of stable parts shared by participants. Then, it is possible to use fewer encryption operations by batching stable parts and balancing the workload between participants rather than encrypting all full-precision parameters by each participant. Since stable parts are encrypted, we can efficiently handle residual parts using a lightweight aggregation method.

The last missing piece of FLSwitch is the design of a learning state-aware switch, toggling between the abovementioned protocols. The crucial question is how to precisely determine the learning state of FL tasks. To tackle the problem, we construct a learning state prediction model based on meta-learning. Since the switch model may yield false predictions, we enable FLSwitch to switch bidirectionally. Therefore, if the switch model detects the model converging, FLSwitch will enable the fast SMA protocol; if the deterioration of the global model is detected, FLSwitch will switch back to the HE-based SMA protocol. In this way, the switch model ensures the convergence of FL tasks.

3.2 Homomorphic Aggregation

Residue-Based PHE Scheme (RBPHE). We now introduce a novel HE-based SMA protocol for the exploring phase, where an efficient residue-based PHE scheme RBPHE will be designed. The goal of RBPHE is to pack a batch of fixed-point numbers into one ciphertext within a flexible encoding range, supporting SIMD operations. As shown in Table 1, we compare the recent HE-based SMA solutions. We consider the overflow and quantization of gradients in real training scenarios, where the gradients follow the nearly Gaussian distribution [4,55]. BatchCrypt [55] reserves enough bits according to the number of parties to avoid overflow. Thus, the solution is limited to scenarios where the addition number should be predefined. FLASHE [22] uses a stateful symmetric scheme and assumes a threat model where the aggregation server does not collude with any party. As for FHE-based solutions, such as CKKS [8], can reach the lowest encryption overhead. However, the ciphertext size is much larger than others, which sacrifices large memory and communication overhead.

Compared with the existing HE-based SMA schemes in Table 1, RBPHE advances in two aspects. On one side, RBPHE takes less amortized overhead for critical operations. On the other side, RBPHE provides an automatic and flexible encoding range extension method, balancing parameter precision and efficiency. In this way, RBPHE achieves a more efficient batching than the existing schemes. Moreover, since RBPHE avoids gradient clipping through dynamic range extension, more precise model updates will be preserved.

Table 1. Comparison of HE-based SMA solutions.

Scheme	Scenarios	Type	Amortized encrypting overhead	Ciphertext Size	Without additive overflow	Without quantization	Weight Multiplication
Paillier	Asymmetric	PHE	High	Large	Yes	Yes	Yes
Naive Batching Paillier [3]	Asymmetric	PHE	Middle	Small	No	No	No
BatchCrypt [55]	Asymmetric	PHE	Middle	Small	Limited	No	No
FLASHE [22]	Symmetric	PHE	Low	Small	Yes	Yes	No
FAHE [9]	Asymmetric	PHE	Low	Large	Yes	Yes	No
CKKS [8]	Asymmetric	FHE	Low	Large	Yes	Yes	Yes
RBPHE	Asymmetric	PHE	Middle	Small	Yes	Yes	Yes

Generally, RBPHE utilizes a residue number system (RNS) to encode a batch of parameters. Specifically, real numbers are converted to residues with predefined prime modulus so that multiple residues can be encoded into a large integer and referred to as a package through the Chinese Remainder Theorem (CRT). We use two moduli (with a particular condition) for each real number and an extra pair of prime moduli to count the addition operations within batching. Therefore, each batch of real numbers will be converted into two packages by RBPHE. Instead of directly encrypting two packages using PHE, a random mask is used to randomize one package and then be encoded to the other package. In this way, we only encrypt the unmasked package to reduce the number of homomorphic operations and improve efficiency.

Intuitively, homomorphic addition will lead to an overflow when the aggregated residue of two encoded real numbers is larger than its prime modulus. To correctly decode addition results beyond the encoding range, we leverage the observation that a small decoding difference will be generated when the sum of two residues is larger than their modulus. Since each real number is encoded using two moduli, a unit difference between a pair of moduli will be detected whenever the encoded residue grows larger than its modulus. Therefore, we can eliminate the effect of overflow by counting unit differences and recovering the original real number. We now present a practical implementation of RBPHE.

$\underline{Setup(\lambda, \mathcal{L}, \mathcal{T}, \mathcal{B})}$: The algorithm takes as input a security parameter λ, an encoding bit length $\mathcal{L}$, a maximum addition time $\mathcal{T}$, and a batching size $\mathcal{B}$, and outputs public parameters $\{\mathcal{P}, \mathcal{Q}\}$ and (pk, sk).

1. Set $\mathcal{T}' = \mathcal{T} \cdot 2^{\lambda}$. Pick two primes $p_0 > \mathcal{T}', q_0 > \mathcal{T}$ and two set of primes $\{p_1, p_2, ..., p_{\mathcal{B}}\}, \{q_1, q_2, ..., q_{\mathcal{B}}\}$ satisfying $p_i > 2^{\mathcal{L}}$ and $(q_i - 1) = k_i(p_i - 1)$, where integer $k_i > 1$.

2. Run $PSetup(\lambda)$ to obtain (pk, sk).
3. Set $\mathcal{P} = \{p_i | 0 \le i \le \mathcal{B}\}, \mathcal{Q} = \{q_i | 0 \le i \le \mathcal{B}\}$ and output $(pk, sk, \{\mathcal{P}, \mathcal{Q}\})$.

$\underline{Encrypt(R, \{\mathcal{P}, \mathcal{Q}\}, pk, \boldsymbol{\theta})}$: The algorithm takes as input an encoding range R, parameters $\boldsymbol{\theta}$, and $\{\mathcal{P}, \mathcal{Q}\}, pk$, and outputs a ciphertext c, if $|\boldsymbol{\theta}| \le \mathcal{B}$. Otherwise, it outputs $\perp$.

- If $|\boldsymbol{\theta}| > |\mathcal{P}|$, directly output $\perp$. Otherwise, pick a mask $r \overset{\$}{\leftarrow} \{0,1\}^\lambda$ uniformly at random and convert each $\theta_i \in \boldsymbol{\theta}$ to residues with $p_i \in \mathcal{P}, q_i \in \mathcal{Q}$ by the following two equations (where $\theta_i = 0$ for $i > |\boldsymbol{\theta}|$):

$$\langle\theta_i\rangle_{p_i} = \left\lceil \frac{\theta_i + R}{2R} \cdot (p_i - 1) \right\rfloor + r \cdot \frac{p_i - 1}{2} \ (mod\ p_i),$$

$$\langle\theta_i\rangle_{q_i} = \left\lceil \frac{\theta_i + R}{2R} \cdot (q_i - 1) \right\rfloor + r \cdot \frac{q_i - 1}{2} \ (mod\ q_i).$$

- Set $\langle\theta_0\rangle_{p_0} = r, \langle\theta_0\rangle_{q_0} = 1, \mathcal{R}_1 = \{\langle\theta_i\rangle_{p_i} | 0 \le i \le \mathcal{B}\}, \mathcal{R}_2 = \{\langle\theta_i\rangle_{q_i} | 0 \le i \le N\}$ and evaluate $\mu_1 \leftarrow crt(\mathcal{P}, \mathcal{R}_1),\ \mu_2 \leftarrow crt(\mathcal{Q}, \mathcal{R}_2)$, where crt is the abstracted function of CRT.
- Run $c_1 \leftarrow PEncrypt(pk, \mu_1)$ and output $c = (c_1, \mu_2)$.

$\underline{Decrypt(\{\mathcal{P}, \mathcal{Q}\}, sk, c)}$: The algorithm takes as input a ciphertext c and $\{\mathcal{P}, \mathcal{Q}\}$,$sk$, and outputs parameters $\boldsymbol{\theta}$, if sk and c are valid. Otherwise, it outputs $\perp$.

- Parse $c = (c_1, \mu_2)$. If $PDecrypt(sk, c_1)$ outputs $\perp$, the algorithm directly outputs $\perp$. Otherwise, it obtains $\mu_1 \leftarrow PDecrypt(sk, c_1)$.
- For $i \leftarrow 1$ to N:
  1. Compute $\langle\theta_i\rangle_{p_i} = \mu_1 - r \cdot \frac{p_i - 1}{2} (mod\ p_i), \langle\theta_i\rangle_{q_i} = \mu_2 - r \cdot \frac{q_i - 1}{2} (mod\ q_i), k_i = \frac{q_i - 1}{p_i - 1}$.
  2. If $\langle\theta_i\rangle_{q_i} < \langle\theta_i\rangle_{p_i} \cdot k$, set $\langle\theta_i\rangle_{q_i} = \langle\theta_i\rangle_{q_i} + q_i$.
  3. Compute unit difference $unit_i = k_i - 1$ and overflow time $t_i = \frac{\langle\theta_i\rangle_{q_i} - k_i \cdot \langle\theta_i\rangle_{p_i}}{k_i - 1}$.
  4. Recover the value $\theta_i = (\langle\theta_i\rangle_{p_i} + t \cdot p - M \cdot \frac{p-1}{2}) \cdot \frac{2}{p-1}$.
- Output $\{\theta_i | 1 \le i \le \mathcal{B}\}$.

$\underline{Add(c, c')}$: The algorithm takes as input two ciphertexts c and c' and outputs a new ciphertext c_{Add}, satisfying $Decrypt(sk, c_{Add}) = Decrypt(sk, c) + Decrypt(sk, c')$.

- Parse $c = (c_1, \mu_2), c' = (c_1', \mu_2')$.
- Evaluate $c_{Add} \leftarrow PAdd(c_1, c_1')$ and $\mu_{Add} = \mu_2 + \mu_2'$.
- Output (c_{Add}, μ_{Add}).

The proposed RBPHE inherits the homomorphic addition algorithm Add from PHE by leveraging the homomorphism of RNS. In particular, a basic prime pair $\langle p_0, q_0\rangle$ is used as a counter of addition operations for the encoded elements. All encoded elements and their addition times will be added when performing element-wise addition on two ciphertexts for the correctness of decoding. To correctly decode the addition results beyond the encoding range $[-R, R]$, resolving the overflow issue, we leverage the observation that a small decoding difference will be generated when the sum of two residues is larger than their modulus. Since each real number is encoded with two modulus, a unit difference between a pair of modulus p_i, q_i will be detected whenever the encoded residue grows larger than its modulus. Therefore, we can eliminate the effect of overflow by counting the number of unit differences and recovering the original real number.

The RBPHE-based SMA protocol is given in Algorithm 1. Before the FL task begins, all participants will invoke the *Setup* algorithm to agree on the same RBPHE instance. During training, the local model of each participant will be encrypted using *Encrypt*. Then the PS collects the encrypted updates and aggregates them through homomorphic addition *Add*. After that, the aggregation result in ciphertext will be sent back to participants. Finally, each participant can learn the aggregation result by *Decrypt*.

Algorithm 1: RBPHE based SMA protocol.

```
Input  : learning rate $\eta$, amount of participants $N$, maximal iteration $T$,
         security parameter $\lambda$, encoding length $\mathcal{L}$, maximum addition time $\mathcal{T}$,
         batching size $\mathcal{B}$, encoding range $R$.
Output: global model $\bar{\boldsymbol{\theta}}_1, \bar{\boldsymbol{\theta}}_2, \ldots, \bar{\boldsymbol{\theta}}_T$.
Initialization:
$(pk, sk, \{\mathcal{P}, \mathcal{Q}\}) \leftarrow Setup(\lambda, \mathcal{L}, \mathcal{T}, \mathcal{B})$
$\bar{\boldsymbol{\theta}}_0 \xleftarrow{\$} (0, 1)$,  $J \leftarrow \lceil \frac{|\bar{\boldsymbol{\theta}}_0|}{\mathcal{B}} \rceil$
Participants:
for $t \leftarrow 1$ to $T$ do
    for $i \leftarrow 1$ to $N$ do
        receive $\boldsymbol{c}'_{t-1} = \{c'_j | 1 \leq j \leq J\}$ from the PS
        for $j \leftarrow 1$ to $J$ do
            $\bar{\boldsymbol{\theta}}_{t-1}[j\mathcal{B} : (j+1)\mathcal{B}] \leftarrow Decrypt(\{\mathcal{P}, \mathcal{Q}\}, sk, c'_j)$
        $\boldsymbol{\theta}^i_t \leftarrow \frac{1}{N}\bar{\boldsymbol{\theta}}_{t-1} - \eta \boldsymbol{g}^i_t$
        for $j \leftarrow 1$ to $J$ do
            $c^i_j \leftarrow Encrypt(R, \{\mathcal{P}, \mathcal{Q}\}, pk, \boldsymbol{\theta}^i_t[j\mathcal{B} : (j+1)\mathcal{B}])$
        send $\boldsymbol{c}^i = \{c^i_j | 1 \leq j \leq J\}$ to the PS
Parameter Server (PS):
for $t \leftarrow 1$ to $T$ do
    receive $\boldsymbol{c}^i$ from $P_i$, $i \in [1, N]$
    for $j \leftarrow 1$ to $J$ do
        $c'_j \leftarrow c^1_j$
        for $i \leftarrow 2$ to $N$ do
            $c'_j \leftarrow Add(c'_j, c^i_j)$
    send $\boldsymbol{c}'_t = \{c'_j | 1 \leq j \leq J\}$ to participants
```

Encoding Precision and Batching Size. Since each parameter θ_i is encoded using two primes p_i, q_i, satisfying $2^{\mathcal{L}} < p_i < q_i$, the encoding precision is determined by the smaller prime p_i. Assuming that the encoding range is $[-R, R]$, the precision can be implicitly inferred by $\frac{2R}{p_i-1} \leq \frac{R}{2^{\mathcal{L}-1}}$. If the rounding operation $\lceil \cdot \rfloor$ is used to round an input to its nearest integer, then the upper bound of the maximum decoding error should be $\frac{R}{p_i-1} \leq \frac{R}{2^{\mathcal{L}}}$. However, RBPHE can flexibly extend its encoding range when necessary. Intuitively, leveraging the addition times encoded with q_0, the maximum encoding range can be extended to $[-q_0 R, q_0 R]$. In this way, the maximum addition time regarding q_0, p_i, q_i can be computed by $min(q_0, \lfloor \frac{q_i(p_i-1)}{q_i-p_i}) \rfloor$, which is usually large enough for real-world FL applications.

The maximal batching size is determined by the key size of PHE, the maximum addition time $\mathcal{T}$, and the primes used in RBPHE. Assuming $\mathbb{Z}_{\mathcal{K}}$ is the input space of PHE according to the security parameter λ, $\mu_1 \leq \mathcal{K}$ should hold to guarantee the correctness of RBPHE. We can first generate enough, say as $\mathcal{B}'$, primes $\widetilde{\mathcal{P}} = \{p_i | 1 \leq i \leq \mathcal{B}'\}$ under a given bit length $\mathcal{L}$. Note that each prime p_i satisfies the condition that there exist another prime q_i and an integer k_i holding $q_i - 1 = k_i(p_i - 1)$. Besides, p_0 can also be determined by $\mathcal{T} \cdot 2^{\lambda}$. Therefore, the maximal batching size can be determined by increasing $\mathcal{B}$ until $\prod_{i=0}^{\mathcal{B}} p_i \geq \mathcal{K}$. The minimal batching size is related to the security of RBPHE and will be discussed in Sect. 4.

3.3 Fast Aggregation

The fast SMA protocol for the converging phase consists of two steps. The first step is a negotiation of the current parameter states, while the second step is secure aggregation. A full-precision model parameter will be split into an anchor and a residual. The anchor part contains most of a parameter's significant digits, while the residual part contains the rest digits. We note that the length of anchors is related to the security of FLSwitch and will be discussed in Sect. 4.

Anchor Negotiation. The basic idea of the first step is to let participants propose their preferred anchors for all parameters. Then the PS arbitrates and yields the chosen anchors for the global model. For security reasons, participants cannot make proposals in plaintext. Assume that a secure hash function $H(\cdot)$ is available globally and a key pair (pk_i, sk_i) is set up beforehand for each P_i for secure communication. Then the negotiation begins with a global random number generation. Each P_i generates a random number s^i and encrypts it as $\tilde{s}^i_{pk_j} = Enc(pk_j, s^i)$. Then P_i sends the message to P_j, $i, j \in [1, n]$, $i \neq j$. Upon receiving $\tilde{s}^j_{pk_i}$ from other participants, P_i decrypts the message and obtains $s^j = Dec(sk_i, \tilde{s}^j_{pk_i})$. In this way, each participant P_i can calculate a global random number $\bar{s} = \sum_{j=1}^{n} s^j$.

Generally, if we flatten model parameters of participant P_i into a 1-D vector, $vec(\boldsymbol{\theta}^i) = \{\theta^i_1, \theta^i_2, \ldots, \theta^i_M\}$, then the j-th parameter in $vec(\boldsymbol{\theta}^i)$ in the t-th training iteration can be denoted by $\theta^i_{j,t}$, $i \in [1, n]$, $j \in [1, M]$, $M = |vec(\boldsymbol{\theta}^i)|$. When denoting the power as γ, $\theta^i_{j,t}$ can be separated into anchor $a^i_{j,t}$ and residue $r^i_{j,t}$ as $\theta^i_{j,t} = a^i_{j,t} \cdot 10^{-\gamma} + r^i_{j,t}$. As γ increases, the range of anchors will be extended, and vice versa. The choice of γ values will be discussed in detail in the analysis and evaluation sections.

In every iteration, P_i calculates $h^i_j = H(a^i_j \oplus \bar{s})$ and sends h^i_j to the PS as an anchor proposal of a parameter θ_j, where $\oplus$ defines an XOR operation. The PS can find the same anchor value by comparing hash results $\boldsymbol{h}_j = \{h^i_j | i \in [1, N]\}$ and select top-ranked proposals as potential anchor values. After counting the frequency of anchor proposals of θ_j, the PS picks K proposals with top frequency and corresponding participants as valid candidates. In each round of negotiation, the PS finds the maximum common subset $\boldsymbol{s}_R$ of candidates for each parameter θ_j, $j \in [1, M]$. Any participant in $\boldsymbol{s}_R$ is available representatives for a_j, noted as $\hat{P}$. Then the negotiation moves on to the rest parameters. The index of parameters with valid candidates will be allocated into a table $\boldsymbol{V} = \{\boldsymbol{v}^i | i \in [1, N]\}$ and added into a set $\boldsymbol{B}_R$. We note that not all parameters' anchors can be negotiated successfully. For example, all proposals of θ_j are different so that $\boldsymbol{s}_R$ is empty. Therefore, we define a sparse ratio $\varepsilon_R = 1 - |\boldsymbol{B}_R|/M$ to indicate negotiation successful ratio. If ε_R is lower than a predefined sparse ratio ε, then we say anchor negotiation for the global model successes.

Algorithm 2: *anchorK*

Input : anchor proposals $\boldsymbol{h}$, amount of top frequent anchor K, sparse rate ε.
Output: parameter allocation table $\boldsymbol{V}$, representative participants set $\mathbb{P}$.

```
$\varepsilon_R \leftarrow 1, \boldsymbol{B}_R \leftarrow \{\}$
$\forall i \in [1, N], \boldsymbol{v}^i \leftarrow \{\}$
for $j \leftarrow 1$ to $M$ do
    $\boldsymbol{P}^j_R \leftarrow$ participants providing $K$ top frequent values of $\boldsymbol{h}_j$
sort $\boldsymbol{P}_R$ by $|\boldsymbol{P}^j_R|$ in descending order
while $\varepsilon_R > \varepsilon$ do
    $\boldsymbol{s}_R \leftarrow \{\}, \boldsymbol{b}_R \leftarrow \{\}$
    for $j \leftarrow 1$ to $M$ and $j \notin \boldsymbol{B}_R$ do
        if $|\boldsymbol{S}_R \cap \boldsymbol{P}^j_R| > 0$ then
            $\boldsymbol{s}_R \leftarrow \boldsymbol{S}_R \cap \boldsymbol{P}^j_R$
            $\boldsymbol{b}_R \leftarrow \boldsymbol{b}_R + \{j\}$
    if $|\boldsymbol{s}_R| == 0$ then
        break
    randomly choose $\hat{P}$ from $\boldsymbol{s}_R$
    $\boldsymbol{v}^{\hat{P}} \leftarrow \boldsymbol{b}_R, \boldsymbol{B}_R \leftarrow \boldsymbol{B}_R + \boldsymbol{b}_R$
    remove $\hat{P}$ from $\boldsymbol{P}_R$
    $\varepsilon_R \leftarrow 1 - \frac{|\boldsymbol{B}_R|}{M}$
$\mathbb{P} \leftarrow \{i \mid |v^i| > 0, v^i \in \boldsymbol{V}\}$
```

The aim of anchor negotiation is to select relatively few participants to represent the majority of parameters by adjusting the arguments K and ε. The selection of appropriate values for K and ε can be solved by empirical analysis. As hyperparameters, K and ε have limited possible values. Thus, it is easy to choose feasible ε values for a given K and vice versa. For brevity, we summarize the abovementioned anchor negotiation and hyperparameter selection as a procedure *AnchorK* in Algorithm 2, taking as input proposals $\boldsymbol{h}$, K, and ε, outputting $\boldsymbol{V}$ and $\mathbb{P}$ for model aggregation in the next step.

Algorithm 3: Fast SMA protocol

```
Input  : learning rate $\eta$, amount of participants $N$, maximal iteration $T$,
         RBPHE parameters $\lambda, \mathcal{L}, \mathcal{T}, \mathcal{B}, R$, negotiation parameters $K, \varepsilon$.
Output: global model $\bar{\boldsymbol{\theta}}_1, \bar{\boldsymbol{\theta}}_2, \ldots, \bar{\boldsymbol{\theta}}_T$.
Initialization:
$(pk, sk, \{\mathcal{P}, \mathcal{Q}\}) \leftarrow Setup(\lambda, \mathcal{L}, \mathcal{T}, \mathcal{B})$
$\bar{\boldsymbol{\theta}}_0 \xleftarrow{\$} (0,1),\ \ J \leftarrow \lceil \frac{|\bar{\boldsymbol{\theta}}_0|}{\mathcal{B}} \rceil$
Participants:
for $t \leftarrow 1$ to $T$ do
    for $i \leftarrow 1$ to $N$ do
        receive $\boldsymbol{c}'_{t-1} = \{c'_j | 1 \le j \le J\}$, $\bar{\boldsymbol{r}}_{t-1}$
        for $j \leftarrow 1$ to $J$ do
            $\bar{\boldsymbol{a}}_{t-1}[j\mathcal{B} : (j+1)\mathcal{B}] \leftarrow Decrypt(\{\mathcal{P}, \mathcal{Q}\}, sk, c'_j)$
        $\boldsymbol{\theta}^i_t \leftarrow \bar{\boldsymbol{a}}_{t-1} + \bar{\boldsymbol{r}}_{t-1} - \eta \boldsymbol{g}^i_t$
        for $j \leftarrow 1$ to $M$ do
            $a^i_{j,t} + r^i_{j,t} \leftarrow \theta^i_{j,t}$
            send $h^i_{j,t} \leftarrow H_s(a^i_{j,t})$ to the PS
        receive $\boldsymbol{v}^i$
        if $|\boldsymbol{v}^i| > 0$ then
            for $j \in \boldsymbol{v}^i$ do
                $c^i_j \leftarrow Encrypt(R, \{\mathcal{P}, \mathcal{Q}\}, pk, \boldsymbol{a}^i_t[j\mathcal{B} : (j+1)\mathcal{B}])$
            send $\boldsymbol{c}^i_t = \{c^i_j | j \in \boldsymbol{v}^i\}$, $\boldsymbol{r}^i_t$ to the PS
Parameter Server (PS):
for $t \leftarrow 1$ to $T$ do
    receive $\boldsymbol{h}_t = \{h^i_{j,t} | 1 \le i \le N, 1 \le j \le M\}$
    $\boldsymbol{V} \leftarrow AnchorK(\boldsymbol{h}_t, K, \varepsilon)$
    send $\boldsymbol{v}^i$ to $P_i (i \in [1, N])$
    receive $\boldsymbol{c}^i_t, \boldsymbol{r}^i_t$ from $P_i (i \in \mathbb{P})$
    $\boldsymbol{c}'_t \leftarrow \{c^i_j | i \in \mathbb{P}, j \in [1, J]\}$,  $\bar{\boldsymbol{r}}_t \leftarrow \frac{1}{|\mathbb{P}|} \sum_i r$
    send $\boldsymbol{c}'_t$, $\bar{\boldsymbol{r}}_t$ to all participants
```

Parameter Aggregation. After selecting proper participants as the representative for parameters in the negotiation, the PS can assign the uploading job according to the allocation table $\boldsymbol{V}$. The selected participants need to upload the allocated anchors $\boldsymbol{c}_t^i$ in the ciphertext and residues $\boldsymbol{r}_t^i$ in plaintext. The PS recombines the $\boldsymbol{c}_t^i$ according to indexes in $\boldsymbol{V}$ and aggregate $\boldsymbol{r}_t^i$ evenly for global parameters. Compared with the HE-based protocol, anchor negotiation brings the extra computation cost in $O(MNlogK)$ and communication cost $N|h|+|\boldsymbol{\theta}|$, where $|h|$ denotes the range of hash function $H(\cdot)$ and $|\boldsymbol{\theta}|$ denotes the index allocation. In the aggregation, we reduce the communication cost from $N|Encrypt(\boldsymbol{\theta})|$ to $K|Encrypt(\boldsymbol{a})| + K|\boldsymbol{r}|$. Considering the ciphertext is much larger than the plaintext, while $\boldsymbol{\theta}$,$\boldsymbol{a}$ and $\boldsymbol{r}$ having the same length, the accelerative ratio of aggregation is N/K. Moreover, since the anchor negotiation may fail for some parameters, we allow the PS to trade the precision of model updating for the optimal job assignment by adjusting K and ε for $AnchorK$. We can let the PS optimize the uploading job assignment considering constraints, including encryption overhead, updating precision, and bandwidth cost. If we assume that all participants have the same equipment, then the PS expects to balance the workload among all participants uniformly. Besides, given the batching capability of RBPHE, the PS should try to assign anchor uploading jobs in multiples of $\mathcal{B}$ to a single participant. Our fast SMA protocol is presented in Algorithm 3 in detail. Encryption operations are inherited from our RBPHE-based SMA protocol since FLSwitch always initializes an FL task using the HE-based protocol.

3.4 Learning State-Aware Switch

Ideally, we want the FLSwitch to start an FL task with the RBPHE-based protocol and then switch to the fast protocol when the learning goes into a stable converging phase. When the global model performance drops, we want the FLSwitch to switch back to the RBPHE-based protocol since it provides more precise model updates. An intuitive way to find the toggling point is by setting a metric threshold. For instance, we can switch between protocols when the test accuracy is higher or lower than a predefined threshold. This hard decision strategy could be in an offline or online mode. In the offline mode, the PS can observe the threshold by pre-trained tasks and adjust it when facing frequent switching. In the online mode, the PS need to dynamically decide the threshold based on the training loss in every epoch. Thus, we construct the switch model by combining a threshold-based hard-decision strategy and a meta-learning based soft-decision strategy.

Suppose that the PS has learning histories of multiple FL tasks following the same task distribution $p(\mathcal{Q})$, where $\mathcal{Q} = \{\mathcal{D}, \mathcal{L}\}$ is an informal definition of an FL task with dataset $\mathcal{D}$ and loss function $\mathcal{L}$. Historic records of FL tasks can be seen as a set of source tasks drawn from $p(\mathcal{Q})$, denoted by $\boldsymbol{Q}_s = \{\{\mathcal{D}_s^{(i)}, \mathcal{L}_s^{(i)}\}|i \in [1, I]\}$. The corresponding models and metrics of source tasks are denoted by $\boldsymbol{\Theta}_s = \{\boldsymbol{\theta}_s^{(i)}|i \in [1, I]\}$, $\boldsymbol{M}_s = \{\boldsymbol{m}_s^{(i)}|i \in [1, I]\}$, where $\boldsymbol{m}$ includes learning metrics such as loss and accuracy. So far, the hard-decision strategy can get a proper threshold by observing $\boldsymbol{M}_s$ and selecting one or more metrics. However, the soft-decision strategy needs another label set $\boldsymbol{Y}_s = \{\boldsymbol{y}_s^{(i)} | i \in [1, I]\}$ indicating learning states for source tasks in $\boldsymbol{Q}_s$. We note that $\boldsymbol{Y}_s$ can be constructed through semi-supervised learning with a small annotated label set.

Now we give the definition of our meta-learning switch model. Given $\boldsymbol{\Theta}_s = \{\boldsymbol{\theta}_s^{(i)} | i \in [1, I]\}$ and $\boldsymbol{M}_s = \{\boldsymbol{m}_s^{(i)} | i \in [1, I]\}$ of source tasks drawn from $p(\mathcal{Q})$, the meta-learning goal of our switch model is to find $\boldsymbol{\theta}^*$, minimizing meta loss

$$\sum_{i \in [1,I]} \mathcal{L}^{meta}(\boldsymbol{\theta}^*(\boldsymbol{\Theta}_s, \boldsymbol{M}_s), \boldsymbol{Y}_s),$$

$$s.t. \quad \boldsymbol{\theta}_s^{(i)} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_s^{(i)}(\boldsymbol{\theta}, \mathcal{D}_s^{(i)}).$$

Then FLSwitch uses $\boldsymbol{\theta}^*$ as a soft-decision model for a target FL task drawn from $p(\mathcal{Q})$, predicting probabilities of the exploring and converging states. Thus, FLSwitch can use both hard-decision and soft-decision strategies in a hybrid way to determine which SMA protocol should be enabled. We note that the hard-decision strategy can ensure the convergence of FL tasks, while the soft-decision strategy is more optimistic about utilizing fast aggregation. In this way, FLSwitch can achieve the best efficiency under the converging constraint.

4 Security Analysis

Intuitively, RBPHE guarantees the irrecoverability of parameters for SMA since μ_1 is encrypted under PHE and μ_2 is masked with a random value. However, one may still be concerned about the semantic information leaked by RBPHE, e.g., whether μ_2 promotes the advantage of an adversary $\mathcal{A}$ to disclose private information. Therefore, we formally prove the indistinguishability under the chosen-plaintext attack (IND-CPA) of RBPHE. The security game of IND-CPA of RBPHE can be briefly abstracted by the following steps:

1. $\mathcal{A}$ chooses $\boldsymbol{\theta}^0, \boldsymbol{\theta}^1$ for a participant with pk.
2. The participant randomly picks $b \xleftarrow{\$} \{0, 1\}$ and sends $c \leftarrow Encrypt(R, \{\mathcal{P}, \mathcal{Q}\}, pk, \boldsymbol{\theta}^b)$ to $\mathcal{A}$.
3. As long as $\mathcal{A}$ desires, it can further request the ciphertext of any $\boldsymbol{\theta}$ from the participant.
4. $\mathcal{A}$ outputs b' and wins if $b' = b$.

We first focus on μ_2 generated by the *Encrypt* algorithm because it is exposed to the adversary $\mathcal{A}$ directly. $\mathcal{A}$ can decompose μ_2 to the residues $\{\langle\theta_i\rangle_{q_i} | 1 \le i \le N\}$ (and $\langle\theta_0\rangle_{q_0} = 1$) with the modulus $\{q_i | 1 \le i \le N\}$. Since each $\langle\theta_i\rangle_{q_i}$ is masked by $r \cdot \frac{q_i - 1}{2}$ within $\mathbb{Z}_{q_i}$, we claim that there only exist two strategies for $\mathcal{A}$ to win the security game with a non-negligible advantage. Given $\boldsymbol{\theta}^0, \boldsymbol{\theta}^1, \langle\boldsymbol{\theta}\rangle = \{\langle\theta_i\rangle_{q_i} | 1 \le i \le N\}$, $\mathcal{A}$

- computes the difference between each two residues $\langle\theta_i\rangle_{q_i} - \langle\theta_j\rangle_{q_j}$ for any $i \ne j$;
- or solves r with $\boldsymbol{\theta}^0$ and $\langle\boldsymbol{\theta}\rangle$, or $\boldsymbol{\theta}^1$ and $\langle\boldsymbol{\theta}\rangle$.

Theorem 1. *Given $\boldsymbol{\theta} = \{\theta_i | 1 \leq i \leq \mathcal{B}\}$, the advantage for an adversary $\mathcal{A}$ to distinguish $\langle\theta_i\rangle_{q_i} - \langle\theta_j\rangle_{q_j}$ $(i \neq j)$ from the difference $v_i - v_j$ of two random values $v_i \in \mathbb{Z}_{q_i}$ and $v_j \in \mathbb{Z}_{q_j}$ is negligible.*

Proof. For $k \in \{i, j\}$, $\langle\theta_k\rangle_{q_k}$ has the following form:

$$\langle\theta_k\rangle_{q_k} = \widetilde{\theta}_k + r(q_k - 1)/2 \ (mod\ q_k),$$

where $\widetilde{\theta}_k = \lceil \frac{\theta_k + R}{2R} \cdot (q_k - 1) \rfloor$. Since q_k is a prime, $\frac{q_k - 1}{2}$ is a generator of $\mathbb{Z}_{q_k}$. With the knowledge of θ_i and θ_j, the consistency of $\langle\theta_i\rangle_{q_i} - \langle\theta_j\rangle_{q_j}$ can be reduced to the indistinguishability between $\pi_1 = r \cdot \frac{q_i - 1}{2}$ $(mod\ q_i) - r \cdot \frac{q_j - 1}{2}$ $(mod\ q_j)$ and $\pi_2 = v_i - v_j$ of two random values v_i, v_j. Generally, r can be redefined by q_i and q_j:

$$r = a_i \cdot q_i + b_i \ = a_j \cdot q_j + b_j,$$

where $a_i, a_j, b_i, b_j \in \mathbb{Z}$. Therefore, π_1 is statistically identical to $b_i \cdot \frac{q_i - 1}{2}$ $(mod\ q_i) - b_j \cdot \frac{q_j - 1}{2}$ $(mod\ q_j)$. Since r is randomly picked and $q_i \neq q_j$, b_i and b_j are independently random to $\mathcal{A}$. In other words, the adversary $\mathcal{A}$ can hardly distinguish π_1 from π_2.

Theorem 2. *Given $\boldsymbol{\theta} = \{\theta_i | 1 \leq i \leq \mathcal{B}\}$, the advantage for an adversary $\mathcal{A}$ to solve r from $\langle\boldsymbol{\theta}\rangle = \{\langle\theta_i\rangle_{q_i} | 1 \leq i \leq \mathcal{B}\}$ is negligible under the hardness of Hilbert's tenth problem [17].*

Proof. To secure the consistency of r, we prove that $\mathcal{A}$ cannot determine whether there exists a solution of r, or recover r with the following equations in polynomial time:

$$\left\{ \langle\theta_i\rangle_{q_i} = \widetilde{\theta}_i + r \cdot \frac{q_i - 1}{2} \ (mod\ q_i) \middle| 1 \leq i \leq \mathcal{B} \right\},$$

where $\widetilde{\theta}_i = \lceil \frac{\theta_i + R}{2R} \cdot (q_i - 1) \rfloor$. It is equivalent to solve r and n_i from the following equation under the constraint that $n_i \in \mathbb{Z}$.

$$\begin{pmatrix} \frac{q_1 - 1}{2} & q_1 & 0 & \cdots & 0 \\ \frac{q_2 - 1}{2} & 0 & q_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{q_{\mathcal{B}} - 1}{2} & 0 & 0 & \cdots & q_{\mathcal{B}} \end{pmatrix} \begin{pmatrix} r \\ n_1 \\ \vdots \\ n_{\mathcal{B}} \end{pmatrix} = \begin{pmatrix} \widetilde{\theta}'_1 \\ \widetilde{\theta}'_2 \\ \vdots \\ \widetilde{\theta}'_{\mathcal{B}} \end{pmatrix},$$

where $\widetilde{\theta}'_i = \langle\theta_i\rangle_{q_i} - \widetilde{\theta}_i$. Therefore, the advantage of solving r is reduced to solving $\mathcal{B}+1$ integers $(r, n_1, \cdots, n_{\mathcal{B}})$ with the above $\mathcal{B}+1$ Diophantine equations, which is a case of Hilbert's tenth problem under the constraints that $0 \leq r \leq 2^{\lambda}$ and $0 \leq n_i \leq \frac{2^{\lambda}}{q_i}$. Solving the equation has been proved to be NP-complete [15] and it has been proved that the Hilbert tenth problem is undecidable for polynomials with 13 variables [34]. Therefore, we can set a batch size no less than the lower bound 13 to ensure that $\mathcal{A}$ cannot solve r.

Theorem 3. *Assuming Theorem 1 and Theorem 2 hold, RBPHE achieves IND-CPA if PHE achieves IND-CPA.*

Proof. Recall that $(\mathcal{P}, \mathcal{Q}, pk)$ are public parameters generated by the *Setup* algorithm. We construct a probabilistic polynomial-time (PPT) simulator $\mathcal{S}$ as follows. Taking as input $(\mathcal{P}, \mathcal{Q}, pk)$ and a vector $\boldsymbol{\theta}$, $\mathcal{S}$ outputs $\perp$ if $|\boldsymbol{\theta}'| > |\mathcal{P}|$. Otherwise, $\mathcal{S}$ picks $\boldsymbol{v} \xleftarrow{\$} \mathbb{Z}_n^{|\boldsymbol{\theta}|}$ and $\mu' \xleftarrow{\$} \mathbb{Z}_n$ uniformly at random, where $\mathbb{Z}_n$ is the input space of PHE using the same security parameter as RBPHE. Then $\mathcal{S}$ evaluates $c' \leftarrow PEncrypt(pk, \mu')$ and encodes $\boldsymbol{v}$ with $\mathcal{Q}$ to obtain μ_2'. Finally, $\mathcal{S}$ outputs (c', μ_2'). For any encoding range R, we prove the indistinguishability $Encrypt(R, \{\mathcal{P}, \mathcal{Q}\}, pk, \boldsymbol{\theta}) \overset{c}{\approx} \mathcal{S}(\{\mathcal{P}, \mathcal{Q}\}, pk, \boldsymbol{\theta})$ via the following hybrid argument:

Hyb_0. Taking as input $(R, \{\mathcal{P}, \mathcal{Q}\}, pk, \boldsymbol{\theta})$, the algorithm *Encrypt* of RBPHE outputs (c, μ_2).

Hyb_1. Same as Hyb_0 except that the algorithm picks $\mu' \xleftarrow{\$} \mathbb{Z}_n$ instead of encoding $\boldsymbol{\theta}$ to μ_1 with $\mathcal{P}$ and encrypting μ_1 to c. The algorithm evaluates $c' \leftarrow PEncrypt(pk, \mu')$ and outputs (c', μ_2). Since PHE achieves IND-CPA, this hybrid is indistinguishable to Hyb_0.

Hyb_2. Same as Hyb_1 except that the algorithm picks $\boldsymbol{v} \xleftarrow{\$} \mathbb{Z}_n^{|\boldsymbol{\theta}|}$ and encodes $\boldsymbol{v}$ to μ_2' with $\mathcal{Q}$ instead of encoding $\boldsymbol{\theta}$ to μ_2 with $\mathcal{Q}$. Assuming Theorem 2 holds, an PPT adversary can distinguish μ' from μ with a negligible probability. Therefore, this hybrid outputs the view of $\mathcal{S}(\{\mathcal{P}, \mathcal{Q}\}, pk, \boldsymbol{\theta})$ and is statically identical to Hyb_1.

Given the security proof of RBPHE, we can directly conclude the security of RBPHE-based SMA protocol. The security analysis of fast SMA protocol is tricky because a hybrid aggregation approach is used. Intuitively, the underlying security issue of the fast SMA protocol is the split of parameters. Since the anchor part of each parameter is encrypted using RBPHE during the aggregation, potential leakage may only be caused by anchor negotiation and residuals aggregation. Given security guarantees of secure hash functions against attacks like collision attack and length attack, we can ensure no leakage will be caused by anchor negotiation if only the global seed $\bar{s}$ is generated randomly. We recall that $\bar{s}$ is constructed by summing random numbers from all participants. Thus, the randomness of $\bar{s}$ can be secured if at least one participant generated random seed honestly.

The aggregation of residuals discloses limited information to the PS and participants. In the PS's view, $r_{j,t}^i$ of P_i's j-th parameter in $vec(\boldsymbol{\theta}^i)$ in the t-th training iteration is accessible, for any $i \in [1, n]$, $j \in [1, M]$, $t \in [1, T]$. However, it is impossible to recover $\theta_{j,t}^i$ from $r_{j,t}^i$. Assume that $a_{j,t}^i$ and $r_{j,t}^i$ represent d_a and d_r significant digits of $\theta_{j,t}^i$, respectively. Then the leakage of $\theta_{j,t}^i$ caused by $r_{j,t}^i$ will be limited by $\frac{d_r}{d_a+d_r}$. Hence, if we choose d_a large enough, accessing $r_{j,t}^i$ is not meaningful for the PS. In the view of any participant P_i, the anchor part of any parameter can be revealed by anchor negotiation in the first step or anchor broadcasting in the second step. By removing P_i's own residual part from the aggregated residuals, P_i can recover $\bar{r}_{j,t} - r_{j,t}^i$. Since P_i colludes with less than

$n-2$ participants, no $r^{i'}_{j,t}$ will be revealed from $\bar{r}_{j,t} - r^{i}_{j,t}$, $i, i' \in [1, n]$, $i' \neq i$. Even if P_i colludes with $n-3$ participants, the only fact can be determined is that $r^{i'}_{j,t}$ varies in $[0, \bar{r}_{j,t} - r^{i}_{j,t}]$. To identify the whole model of target $P_{i'}$, P_i needs at least $10^{d_r \times M}$ guesses.

5 Evaluation

5.1 Experimental Setup

We have implemented FLSwitch and evaluated our solution comprehensively. All the experiments are performed on a Linux server with Intel(R) Xeon(R) Gold 5115 CPU running at 2.40GHz on 10 cores and 503 GB RAM. We use MNIST [27], FASHION-MNIST [51], CIFAR10, and CIFAR100 [25] datasets. Our first application is a 3-layer fully-connected neural network on MNIST and FASHION-MNIST, having 55050 network parameters in total. The other application is a 20-layer ResNet [16] on CIFAR-10 and CIFAR-100, having 272474 parameters in total. We evaluate FLSwitch in three metrics, model performance, execution time, and communication cost. Besides, we study how key system parameters affect these metrics of FLSwitch, including K, ε, and N. K and ε are crucial to the fast aggregation protocol, while N reflects the solution scalability. The precision power γ is set to the most common power of parameters with one significant digit before the first switch. The experimental result demonstrates that the predefined γ works well in the subsequent learning phase.

Unless otherwise noted, we use the following default settings for evaluation. $N = 10$ for all datasets, $K = 3$ and $\varepsilon = 0.05$ for MNIST and FASHION-MNIST, $K = 1$ and $\varepsilon = 0.01$ for CIFAR10 and CIFAR100. For instance, the first image in the second row of Fig. 2 evaluates the impact of ε with $N = 10$ and $k = 3$ on MNIST. We evaluate the execution time and communication cost of FLSwitch and compare them with the existing solutions, including the original PHE, CKKS, and BatchCrypt [55]. When evaluating homomorphic operations, a 10-participant FL task is used with a 2048-bit key for PHE and a 128-bit security parameter for CKKS. And the comparison of different protocols is conducted using FASHION-MNIST and CIFAR-10. The encrypted data uses 16 bits precision in default.

5.2 Experimental Result

We evaluate the model performance of FLSwitch using model testing accuracy and training loss. The figure matrix in Fig. 2 shows model performance evaluation results on different datasets using various system parameters. In particular, each column of the figure shows results on a single dataset, while each row gives detailed results regarding different system parameters K, ε, and N. Moreover, a baseline model trained on plaintexts is compared with FLSwitch as a reference. It can be concluded from the figure that FLSwitch performs closely to the baseline on MNIST and FASHION-MNIST and even performs better than the baseline in

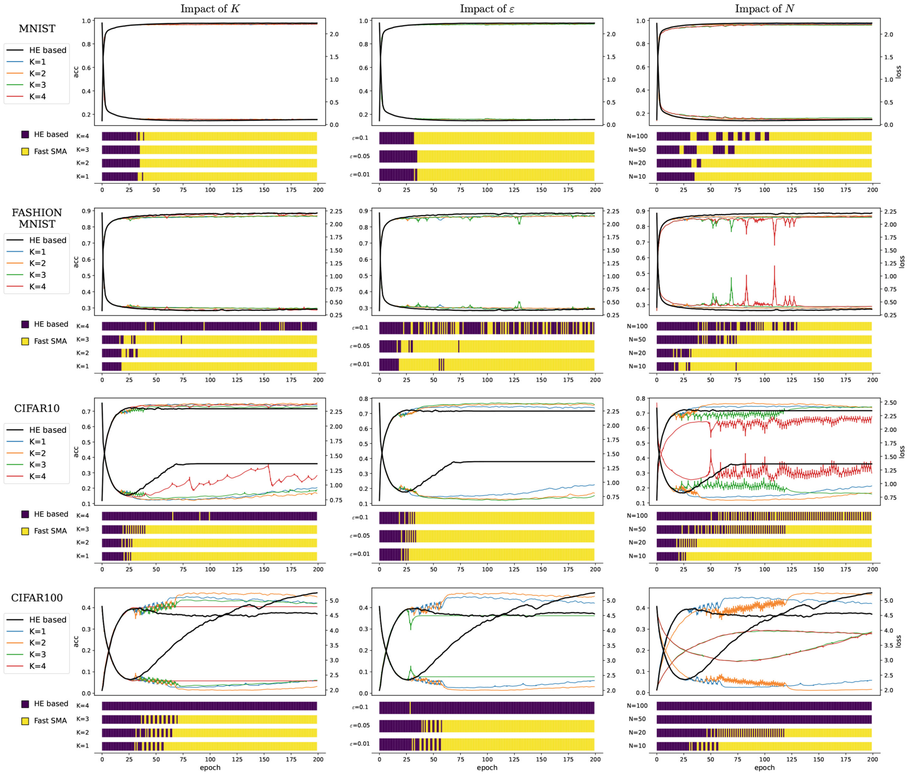


Fig. 2. Evaluation of the global model performance using FLSwitch with different system parameters, including K, ε, and N.

some cases on CIFAR10 and CIFAR100 because the baseline model is too simple to fit the CIFAR100 dataset and results in overfitting. However, the negotiation process of FLSwitch mitigates the overfitting phenomenon of FL tasks, especially for CIFAR100 models. As shown in the first row of Fig. 2, a smaller K performs more stable in accuracy and loss. Meanwhile, it switches less frequently. On the contrary, when K is larger than or equal to 4, FLSwitch almost maintains the HE-based protocol during the whole training process.

Intuitively, K impacts the number of selected participants, and ε impacts the ratio of parameters controlled by these participants. When K is too large, any participant could be selected as the only one who controls all the parameters. Meanwhile, when ε is too large, the selected ones will lose control due to insufficient parameter density. As shown in the first and second columns of Fig. 2, the unsuitable values of K and ε prolongs the fluctuation range of the learning curve and the switching. Since both over-control and under-control cases should be avoided in FLSwitch, we recommend $K = 3$, $\varepsilon = 0.05$ for MNIST and FASHION-MNIST, $K = 1$, $\varepsilon = 0.01$ for CIFAR10 and CIFAR100.

The third column of Fig. 2 shows how the number of participants impacts the model performance under the default K and ε. It can be found that the increasing N causes the more obvious prolongation of the learning curve and the switching time. However, the curve keeps stable when the learning is switched to the fast protocol in the converging phase. On the other side, when N is larger than or equal to 100, FLSwitch prefers to stay with the HE-based protocol because the divergence of participants is significant. This result can be changed by adjusting K and ε for large-scale FL tasks.

We evaluate execution times and communication costs of each participant and the server in Table 2, 3, and 4. Table 2 shows that the RBPHE scheme has a much smaller cipher size than CKKS. Compared with the BatchCrypt scheme, RBPHE supports a larger maximal batch size, meaning more plaintext data can be encoded in one package, leading to a higher compression rate and lower execution overhead. For example, our RBPHE can support a 200 batch size with a 2048-bit key and 16-bit precision. However, the batch size of BatchCrypt could only arrive at approximately 100 in the same setting. Moreover, the RBPHE scheme uses a more flexible addition operation and has overcome the overflow problem, which is a main weakness of the BatchCrypt scheme.

The results in Table 3 and Table 4 show that FLSwitch reduces the total execution time and balances the loads between participants and the server when compared to the prior HE-based schemes. The RBPHE scheme has less execution time than PHE and BatchCrypt but a slightly more communication cost than BatchCrypt. Moreover, the fast SMA protocol (referred to as FastAgg in tables) offloads part of computing loads from participants to the server, reducing the total execution time. That is to say, the fast SMA protocol has fewer encryption operations. However, extra communication rounds in the fast SMA protocol are caused. When the number of participants increases, the total cost of FLSwitch will get close to the PHE. Considering the execution time reduced by FLSwitch, the additional communication cost is acceptable, especially when the server is a resourceful center. Besides, the communication cost of FLSwitch is much better than SMC-based SMA solutions like [7,46].

Table 2. Performance comparison of homomorphic encryption.

Input Size	Scheme *	Size (KB)	Enc (ms)	Dec (ms)	Add (ms)	Mul (ms)
4096	CKKS (16bit)	1280.1	16	6	0.6	1
	BatchCrypt (8bit)	16.6	350	105	1	/
	BatchCrypt (16bit)	25.7	528	157	2	/
	RBPHE (8bit)	19.1	261	79	1	0.8
	RBPHE (16bit)	30.3	411	122	2	1
65536	CKKS (16bit)	10241.1	127	44	5	8
	BatchCrypt (8bit)	256.8	5442	1633	24	/
	BatchCrypt (16bit)	403.3	8275	2452	36	/
	RBPHE (8bit)	297.8	4142	1254	17	13
	RBPHE (16bit)	479.9	6497	1916	28	21

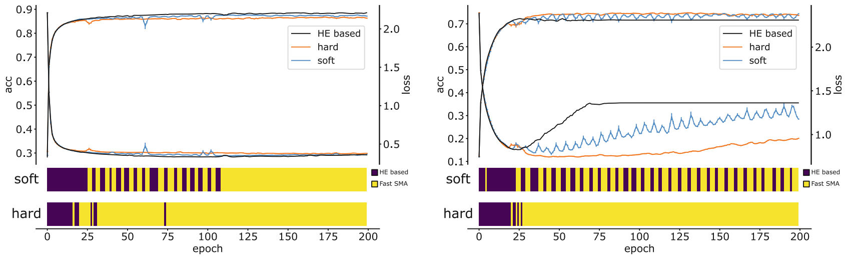
* We use the implementation of CKKS in the SEAL library. The implementation of BatchCrypt and RBPHE is based on python-paillier.

Table 3. Execution time results of HE-based SMA solutions (ms).

Dataset	Protocol	Clients10		Clients50		Clients100	
		Client	Server	Client	Server	Client	Server
FASHION MNIST	**FastAgg**	2.22	2.36	5.03	13.48	28.21	3.79
	RBPHE	8.64	0.19	23.68	1.04	47.15	2.08
	Paillier	8.72	0.07	25.53	0.41	52.69	0.86
	Batchcrypt	9.64	0.26	26.24	1.48	51.83	3.06
CIFAR10	**FastAgg**	24.92	17.91	56.83	26.38	165.74	19.23
	RBPHE	43.17	0.95	122.21	5.12	233.03	10.28
	Paillier	47.10	0.35	130.46	2.24	263.89	4.45
	Batchcrypt	48.59	1.20	131.27	6.65	266.74	14.98

Table 4. Communication cost results of HE-based SMA solutions (MB).

Dataset	Protocol	Clients10		Clients50		Clients100	
		Client	Server	Client	Server	Client	Server
FASHION MNIST	**FastAgg**	0.64	7.88	0.53	33.34	0.48	59.59
	RBPHE	0.37	3.65	0.37	18.26	0.37	36.51
	Paillier	0.28	2.82	0.31	15.56	0.32	32.22
	Batchcrypt	0.29	2.94	0.29	14.68	0.29	29.39
CIFAR10	**FastAgg**	4.02	43.79	2.62	163.82	2.21	260.33
	RBPHE	1.80	18.01	1.80	90.07	1.80	180.1
	Paillier	1.52	15.23	1.68	83.82	1.74	173.68
	Batchcrypt	1.44	14.49	1.44	72.46	1.44	144.86



(a) FASHION-MNIST

(b) CIFAR10

Fig. 3. Prediction result of the state-aware switch model.

We evaluate the effectiveness of the learning state-aware model and give the result in Fig. 3. We can see that the prediction model chooses the RBPHE-based protocol in the exploring phase and switches to the fast aggregation protocol in the converging phase, just as expected. However, we notice that the prediction model may cause switching oscillations, attempting to improve the efficiency by applying the fast aggregation protocol but may get failed several times. We note that the result may be caused by the model's overfitting. For example, FLSwitch tries to improve the CIFAR10 model performance but finds it impossible due to overfitting.

6 Related Work

Numerous research papers have applied HE protections in FL [28] under similar settings to our study. Different security requirements are satisfied by various encryption systems, such as the RSA-based [53], ElGamal-based [10], Paillier-based [11,31], CKKS-based [38] and so on. These solutions mainly focus on the security requirements but ignore the FL learning characteristics, raising efficiency issues.

There also exist research papers devoted to developing efficient and secure aggregation protocols for FL. Device scheduling in training is usually applied to reduce the interaction frequency under limited bandwidth. Recent studies [1,2] restrict the number of scheduled devices based on the channel conditions and the significance of local model updates measured by the l2-norm. Besides, the significance can also be measured by gradient divergence, appointing different scheduled probabilities to devices [42]. Meanwhile, local models in FL can be abstracted into a simplified computational graph based on the salient parameters in the network [54]. The PS, as an agent, takes the graphs as input and produces the selection policy. But the process brings excessive workload to the PS. Unfortunately, the abovementioned studies cannot provide security guarantees for model parameters.

Quantization and sparsification [21] are state-of-the-art methods to reduce communication overhead via compressing the parameters in FL. Quantization limits the number of bits of floating point parameters, especially the gradients. Sparsification only transmits the large enough entries of gradients and drops or accumulates the smaller ones. Specifically, the gradient differences can be compressed via stochastic quantization and sparsification [19,37]. Meanwhile, the redundant gradient updates of small differences after the quantization can be skipped for reduction [47]. However, compared to HE-based FL, the security is inadequate when the local parameters must be exposed to the PS.

Our FLSwitch takes into account different learning phases of FL, aiming to only select the representative local model parameters as scheduled communication participants. The PS is only required to execute a simple alignment task for the selection instead of calculating the comparison of the whole model. The confidentiality of transmitted parameters can be enforced by HE protocols. Other security requirements, such as published model inference resistance [32] and poisoning attack defense [33], should be studied separately.

7 Conclusion

We propose a new HE-based SMA solution by leveraging PHE and a residue number coding system, outperforming the existing work. Besides, we give the first attempt to further improve SMA efficiency by utilizing FL characteristics, which significantly reduces the overhead per participant. We note that FLSwitch is designed for data confidentiality, which means that we exclude poisoning attacks [12,33] against parameters or anchors. However, data poisoning or backdoor attacks that indirectly interfere with the global model may also affect FLSwitch and should be investigated further. Future work includes exploring the use of meta-learning model and improving the scalability of our scheme. We hope the meta-learning model can make the decision more stably according to detailed performance measurements. Additionally, we will expand our scheme to other domains such as finance and healthcare datasets.

Acknowledgement. The authors would like to thank the anonymous reviewers for the time and efforts they have kindly made in this paper. This work was supported in part by the National Key R&D Program of China under Grants 2020YFB1005900, the Leading-edge Technology Program of Jiangsu-NSF under Grant BK20222001 and BK20202001, the National Natural Science Foundation of China under Grants NSFC-62272222, NSFC-61902176, NSFC-62272215.

References

1. Amiri, M.M., Gündüz, D., Kulkarni, S.R., Poor, H.V.: Update aware device scheduling for federated learning at the wireless edge. In: 2020 IEEE International Symposium on Information Theory (ISIT), pp. 2598–2603. IEEE (2020)
2. Amiri, M.M., Gündüz, D., Kulkarni, S.R., Poor, H.V.: Convergence of update aware device scheduling for federated learning at the wireless edge. IEEE Trans. Wireless Commun. **20**(6), 3643–3658 (2021)
3. Aono, Y., Hayashi, T., Wang, L., Moriai, S., et al.: Privacy-preserving deep learning via additively homomorphic encryption. IEEE Trans. Inf. Forensics Secur. **13**(5), 1333–1345 (2017)
4. Baskin, C., et al.: UNIQ: uniform noise injection for non-uniform quantization of neural networks. ACM Trans. Comput. Syst. (TOCS) **37**(1–4), 1–15 (2021)
5. Bell, J.H., Bonawitz, K.A., Gascón, A., Lepoint, T., Raykova, M.: Secure single-server aggregation with (poly) logarithmic overhead. In: ACM SIGSAC Conference on Computer and Communications Security, pp. 1253–1269 (2020)
6. Bonawitz, K., et al.: Towards federated learning at scale: system design. Proc. Mach. Learn. Syst. **1**, 374–388 (2019)
7. Bonawitz, K., et al.: Practical secure aggregation for privacy-preserving machine learning. In: ACM SIGSAC Conference on Computer and Communications Security, pp. 1175–1191 (2017)
8. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 409–437. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_15
9. Cominetti, E.L., Simplicio, M.A.: Fast additive partially homomorphic encryption from the approximate common divisor problem. IEEE Trans. Inf. Forensics Secur. **15**, 2988–2998 (2020)
10. Fang, C., Guo, Y., Hu, Y., Ma, B., Feng, L., Yin, A.: Privacy-preserving and communication-efficient federated learning in internet of things. Comput. Secur. **103**, 102199 (2021)
11. Fang, H., Qian, Q.: Privacy preserving machine learning with homomorphic encryption and federated learning. Future Internet **13**(4), 94 (2021)

12. Fang, M., Cao, X., Jia, J., Gong, N.: Local model poisoning attacks to {Byzantine-Robust} federated learning. In: USENIX Security Symposium, pp. 1605–1622 (2020)
13. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning, pp. 1126–1135 (2017)
14. Guo, X., et al.: VeriFL: communication-efficient and fast verifiable aggregation for federated learning. IEEE Trans. Inf. Forensics Secur. **16**, 1736–1751 (2020)
15. Gurari, E.M., Ibarra, O.H.: An NP-complete number-theoretic problem. J. ACM (JACM) **26**(3), 567–581 (1979)
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
17. Hilbert, D.: Mathematische probleme. In: Dritter Band: Analysis · Grundlagen der Mathematik · Physik Verschiedenes, pp. 290–329. Springer, Berlin (1935). https://doi.org/10.1007/978-3-662-38452-7_19
18. Hitaj, B., Ateniese, G., Perez-Cruz, F.: Deep models under the GAN: information leakage from collaborative deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 603–618 (2017)
19. Horváth, S., Kovalev, D., Mishchenko, K., Richtárik, P., Stich, S.: Stochastic distributed learning with gradient quantization and double-variance reduction. Optim. Methods Softw., 1–16 (2022)
20. Huang, Y., Gupta, S., Song, Z., Li, K., Arora, S.: Evaluating gradient inversion attacks and defenses in federated learning. In: Advances in Neural Information Processing Systems, vol. 34 (2021)
21. Jiang, P., Agrawal, G.: A linear speedup analysis of distributed deep learning with sparse and quantized communication. In: Advances in Neural Information Processing Systems, vol. 31 (2018)
22. Jiang, Z., Wang, W., Liu, Y.: FLASHE: additively symmetric homomorphic encryption for cross-silo federated learning. arXiv preprint: arXiv:2109.00675 (2021)
23. Kaya, Y., Dumitras, T.: When does data augmentation help with membership inference attacks? In: International Conference on Machine Learning, pp. 5345–5355 (2021)
24. Krause, A., Guestrin, C.: Nonmyopic active learning of gaussian processes: an exploration-exploitation approach. In: International Conference on Machine Learning, pp. 449–456 (2007)
25. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
26. Lai, F., Zhu, X., Madhyastha, H.V., Chowdhury, M.: Oort: efficient federated learning via guided participant selection. In: USENIX Symposium on Operating Systems Design and Implementation, pp. 19–35 (2021)
27. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
28. Liu, Z., Guo, J., Yang, W., Fan, J., Lam, K.Y., Zhao, J.: Privacy-preserving aggregation in federated learning: a survey. IEEE Trans. Big Data (2022)
29. Luo, B., Li, X., Wang, S., Huang, J., Tassiulas, L.: Cost-effective federated learning design. In: IEEE Conference on Computer Communications, pp. 1–10 (2021)
30. Luo, X., Wu, Y., Xiao, X., Ooi, B.C.: Feature inference attack on model predictions in vertical federated learning. In: International Conference on Data Engineering (ICDE), pp. 181–192 (2021)

31. Ma, J., Naas, S.A., Sigg, S., Lyu, X.: Privacy-preserving federated learning based on multi-key homomorphic encryption. Int. J. Intell. Syst. **37**(9), 5880–5901 (2022)
32. Mao, Y., Hong, W., Zhu, B., Zhu, Z., Zhang, Y., Zhong, S.: Secure deep neural network models publishing against membership inference attacks via training task parallelism. IEEE Trans. Parallel Distrib. Syst. **33**(11), 3079–3091 (2021)
33. Mao, Y., Yuan, X., Zhao, X., Zhong, S.: Romoa: robust Model Aggregation for the resistance of federated learning to model poisoning attacks. In: Bertino, E., Shulman, H., Waidner, M. (eds.) Computer Security—ESORICS 2021. ESORICS 2021. Lecture Notes in Computer Science(), vol. 12972, pp. 476–496 . Springer, Cham. https://doi.org/10.1007/978-3-030-88418-5_23
34. Matijasevič, Y., Robinson, J.: Reduction of an arbitrary Diophantine equation to one in 13 unknowns. **6**, 235 (1996). The Collected Works of Julia Robinson
35. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282 (2017)
36. Melis, L., Song, C., De Cristofaro, E., Shmatikov, V.: Exploiting unintended feature leakage in collaborative learning. In: IEEE Symposium on Security and Privacy (SP), pp. 691–706 (2019)
37. Mishchenko, K., Gorbunov, E., Takáč, M., Richtárik, P.: Distributed learning with compressed gradient differences. arXiv preprint: arXiv:1901.09269 (2019)
38. Mouchet, C., Troncoso-Pastoriza, J.R., Hubaux, J.P.: Multiparty homomorphic encryption: from theory to practice. IACR Cryptol. ePrint Arch. **2020**, 304 (2020)
39. Nasr, M., Shokri, R., Houmansadr, A.: Comprehensive privacy analysis of deep learning: passive and active white-box inference attacks against centralized and federated learning. In: IEEE Symposium on Security and Privacy (SP), pp. 739–753 (2019)
40. Nguyen, H.T., Sehwag, V., Hosseinalipour, S., Brinton, C.G., Chiang, M., Poor, H.V.: Fast-convergent federated learning. IEEE J. Sel. Areas Commun. **39**(1), 201–218 (2020)
41. Pasquini, D., Ateniese, G., Bernaschi, M.: Unleashing the tiger: inference attacks on split learning. In: ACM SIGSAC Conference on Computer and Communications Security, pp. 2113–2129 (2021)
42. Ren, J., He, Y., Wen, D., Yu, G., Huang, K., Guo, D.: Scheduling for cellular federated edge learning with importance and channel awareness. IEEE Trans. Wireless Commun. **19**(11), 7690–7703 (2020)
43. Sav, S., et al.: POSEIDON: privacy-preserving federated neural network learning. In: Network and Distributed System Security Symposium, NDSS (2021)
44. Shokri, R., Shmatikov, V.: Privacy-preserving deep learning. In: ACM SIGSAC Conference on Computer and Communications Security, pp. 1310–1321 (2015)
45. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: IEEE Symposium on Security and Privacy (SP), pp. 3–18 (2017)
46. So, J., Güler, B., Avestimehr, A.S.: Turbo-aggregate: breaking the quadratic aggregation barrier in secure federated learning. IEEE J. Sel. Areas Inf. Theory **2**(1), 479–489 (2021)
47. Sun, J., Chen, T., Giannakis, G.B., Yang, Q., Yang, Z.: Lazily aggregated quantized gradient innovation for communication-efficient federated learning. IEEE Trans. Pattern Anal. Mach. Intell. **44**(4), 2031–2044 (2020)
48. Sun, L., Qian, J., Chen, X.: LDP-FL: practical private aggregation in federated learning with local differential privacy. In: International Joint Conference on Artificial Intelligence, IJCAI, pp. 1571–1578 (2021)

49. Wang, Z., Song, M., Zhang, Z., Song, Y., Wang, Q., Qi, H.: Beyond inferring class representatives: User-level privacy leakage from federated learning. In: IEEE Conference on Computer Communications, pp. 2512–2520 (2019)
50. Wei, K., et al.: Federated learning with differential privacy: algorithms and performance analysis. IEEE Trans. Inf. Forensics Secur. **15**, 3454–3469 (2020)
51. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint: arXiv:1708.07747 (2017)
52. Xu, G., Li, H., Liu, S., Yang, K., Lin, X.: VerifyNet: secure and verifiable federated learning. IEEE Trans. Inf. Forensics Secur. **15**, 911–926 (2019)
53. Yang, W., Liu, B., Lu, C., Yu, N.: Privacy preserving on updated parameters in federated learning. In: Proceedings of the ACM Turing Celebration Conference-China, pp. 27–31 (2020)
54. Yu, S., Nguyen, P., Abebe, W., Qian, W., Anwar, A., Jannesari, A.: SPATL: salient parameter aggregation and transfer learning for heterogeneous federated learning. In: 2022 SC22: International Conference for High Performance Computing, Networking, Storage and Analysis (SC), pp. 495–508. IEEE Computer Society (2022)
55. Zhang, C., Li, S., Xia, J., Wang, W., Yan, F., Liu, Y.: BatchCrypt: efficient homomorphic encryption for cross-silo federated learning. In: USENIX Annual Technical Conference, pp. 493–506 (2020)
56. Zhang, W., Tople, S., Ohrimenko, O.: Leakage of dataset properties in {Multi-Party} machine learning. In: USENIX Security Symposium, pp. 2687–2704 (2021)
57. Zheng, Q., Chen, S., Long, Q., Su, W.: Federated f-differential privacy. In: International Conference on Artificial Intelligence and Statistics, pp. 2251–2259 (2021)