FedSSA: Reducing Overhead of Additive Cryptographic Methods in Federated Learning with Sketch

Paper ID: # 465

***Abstract*—Federated Learning (FL) has been applied across diverse domains as a powerful technique but faces critical challenges in privacy protection. *Secure aggregation* and *additive homomorphic encryption* are two of the most commonly used cryptographic methods to protect model updates. To provide a strong privacy guarantee, both methods satisfy the additivity and require the integer form to encrypt model updates. However, they still suffer from a non-negligible overhead of the computation and communication. To mitigate such overhead, existing approaches of lossy compression (e.g., gradient compression) have been explored but exhibit the inapplicability for FL with additive cryptographic methods. In this paper, we propose FEDSSA, a novel compression framework to reduce the overhead of additive cryptographic methods in FL based on two new techniques: (i) QSRHT Sketch, a sketch-based compression method which supports large compression ratios with a bounded error with the integer requirement, and (ii) periodic rehashing, which ensures the unbiasedness of QSRHT Sketch. Our evaluation shows that FEDSSA achieves a high compression ratio ($\times$160) with a low model accuracy degradation (less than 5%). In the additive homomorphic encryption, FEDSSA reduces the average computation time per round by up to 58.15% compared to state-of-the-art compressors that support additive homomorphic encryption, with a low test accuracy drop (less than 2.2%).**

I. INTRODUCTION

Federated Learning (FL) is a machine learning paradigm to maintain data privacy via training models on decentralized data sources. It has been widely applied in various domains, including intelligent personal assistants [1], healthcare [2], and financial support [3]. A typical training procedure of FL has two parties [4], i.e., a *logically centralized server* (server for short) and multiple *decentralized clients* (clients for short). The training procedure iterates for multiple rounds. In each round, clients train a shared model using their local data and send only *model updates* (usually gradients) to the server, instead of raw data as in traditional machine learning.

The concern of privacy leakage is crucial in FL. Despite clients keeping their local data away from the server, the server can still infer sensitive information from individual model updates. Specifically, the server can reconstruct the client's sensitive data from individual model updates through data reconstruction attacks or gradient inversion attacks [5]–[8]. This poses a privacy vulnerability in the client's local dataset. To this end, a substantial body of existing research has tackled this issue with privacy-preserving methods using cryptographic schemes [9]–[12]. These methods are provably secure and can provide a strong privacy guarantee. Among them, *secure aggregation* [10], [13], [14] and *additive homomorphic encryption* [11], [15], [16] are two of the most commonly used methods. Both methods are referred to as the *additive cryptographic methods*, where the additivity means the sum operation can be directly applied to encrypted model updates and yield meaningful aggregated results. Such additive cryptographic methods execute encryption and decryption procedures in integer fields, meaning that the plaintext (i.e. model updates) should be in the integer form [15], [17].

While additive cryptographic methods provide strong privacy guarantees, they still incur significant overhead during computational and communication procedures. Specifically, computational overhead in these methods primarily results from complex modular operations during encryption or decryption processes [16], while the communication overhead is mainly caused by the slow communication connection between clients and the server. Even worse, secure aggregation and additive homomorphic encryption cause the *data inflation* problem, where the ciphertext may be larger than the raw model updates [16], [17]. This further exacerbates the overhead and slows down the FL training procedure.

To mitigate the overhead of computation and communication in additive cryptographic methods, existing studies [18]–[22] have explored the *gradient compression* to reduce the size of model updates in FL. However, gradient compression approaches exhibit the following limitations which renders them unsuitable for FL with additive cryptographic methods. First, many of the existing methods lack additivity, which renders them incompatible with additive cryptographic methods [18]–[20], [23], [24]. Second, certain methods only support a low compression ratio [16], [25]. Therefore, these methods can only offer a limited overhead reduction for additive cryptographic methods. Third, some of the gradient compression methods generate biased model updates, which can compromise the final model accuracy [26].

In this paper, we propose FEDSSA, which adopts sketch-based compression method to address the above limitations. Sketches belong to a family of randomized data structures that compress large datasets using compact arrays and random hash functions. We adopt the *Subsampled Randomized Hadamard Transform* (SRHT) sketch [27], [28], a well-established technique in traditional machine learning. SRHT Sketch is additive and supports a large compression ratio with bounded error. However, SRHT Sketch faces challenges in compatibility with additive cryptographic methods for two main reasons. First, to meet the integer requirement of additive cryptographic methods, SRHT Sketch should convert its fractional elements into integers with bounded error. When compressing model updates, SRHT Sketch takes fractional model updates as input and generates fractional elements. Converting each fractional

element into an integer using operations like floor or ceiling functions may seem straightforward. However, these methods may introduce unbounded errors due to the precision loss of directly discarding the fractional parts. Second, it is important to efficiently refresh hash functions to guarantee independence. SRHT Sketch relies on the independence of these hash functions to generate unbiased model updates in each round. While refreshing the hash functions can maintain independence, it is challenging to do so efficiently and correctly. The clients can actively send requests for the latest hash functions. However, this would cause additional processing overhead on the server side when multiple clients send requests simultaneously. Also, it is crucial to update hash functions correctly on the client side. Updating the hash function before the client decompresses with the old ones leads to incorrect model updates and introduces errors in the FL procedure.

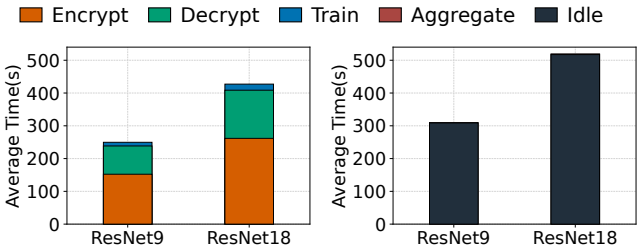
FEDSSA overcomes the above challenges with two new techniques. For the first challenge, FEDSSA adopts QSRHT Sketch, which enhances SRHT Sketch [27], [28] with a quantization step. In this step, QSRHT Sketch maps each fractional element to an integer with unbiased estimation and a bounded error. For the second challenge, FEDSSA proposes an efficient protocol called *periodic rehashing* that refreshes the hash functions of QSRHT Sketch every round and maintains the correctness of hash function usage. We conduct a comprehensive evaluation by comparing FEDSSA to the six state-of-the-art baselines including two secure aggregation-based compressors [29], one additive homomorphic encryption compressor [16], and three gradient-based compressors [22], [30], [31]. Our evaluation of two training tasks (i.e., ResNet9 [32] on CIFAR-10 and ResNet18 [33]on CIFAR-100) shows that FEDSSA achieves a high compression ratio of up to $\times$160 with no more than 5% model accuracy degradation. When comparing to baseline compressors, FEDSSA achieves comparable test accuracy under the same compression ratio. Moreover, FEDSSA with a larger compression ratio can reduce the average computation time per round by up to 58.15% compared to the additive homomorphic encryption-based compressor, with a test accuracy drop of no more than 2.2%.

II. BACKGROUND AND MOTIVATION

A. Federated Learning

Federated Learning (FL) is a collaborative machine-learning approach including two parties: a centralized server and multiple decentralized clients [4]. The server coordinates the training of a global model among clients with multiple rounds. In each round, each client trains a local model using its local data and sends *model updates* (i.e., gradients) to the server for aggregation. The server aggregates local models into the global model and sends them back to clients for training in the next round. The process iterates until the desired number of rounds or the target model accuracy is achieved.

Privacy leakage and threat model. Despite clients keeping their local data away from the server, the server can still infer sensitive information from individual model updates [5]–[8]. For example, it can accurately reconstruct a client's training data through a *data reconstruction attack* based on gradient updates. In this paper, we consider a threat model where both the server and clients are *honest-but-curious*, which is standard in the context of FL [5], [10], [16], [34]. In this model, the server faithfully aggregates model updates from clients, but it attempts to infer the private information of each client. Clients execute local training faithfully and do not collude with the server or other clients. Although some existing works consider the potential privacy leakage caused by a client dropout or straggler [34], [35], in our threat model, we do not consider these scenarios as most of these situations can be managed by off-the-shelf privacy protection mechanisms [10]. We leave the exploration of these issues for future work.



(a) Client-side (b) Server-side

Figure 1: Average time per round breakdowns for client and server.

B. Secure Aggregation and Additive Homomorphic Encryption

To protect the privacy, *secure aggregation* [10], [13], [14] and *additive homomorphic encryption* [11], [15], [16] are widely used cryptographic methods to ensure the confidentiality of individual model updates. They require the following properties in use.

- **Additivity.** The property ensures that the additive operation can be directly applied to the encrypted model updates for generating aggregated results. Also, decrypting the aggregated results yields the corresponding aggregated model updates. Thus, the plaintext (i.e., model updates) should also be additive. Without additivity, it is impossible to obtain aggregated model updates by decrypting the aggregated results.
- **Integer arithmetic.** Both methods perform encryption and decryption procedures on the integer field. Thus, the model updates should be in integer form.

Secure aggregation. Secure aggregation (SA) is a widely employed privacy protection method in FL [10], [13], [14]. In each round, SA involves clients generating cancellable integer masks that mutually cancel out when summed. After that, clients encrypt their model updates by adding up the corresponding masks to obtain ciphertexts and send the ciphertexts to the server. The server then directly sums the ciphertexts to obtain aggregated model updates without decrypting individual contributions. This aggregation is feasible, since when the ciphertexts are summed, the integer masks of each party nullify one other, resulting in the aggregated sum identical to that of model updates. This procedure ensures that the server cannot discern individual model updates beyond their aggregated sum.

Additive homomorphic encryption. Additive homomorphic encryption (AHE) is a variant of homomorphic encryption with supporting the additive operation on ciphertexts [15], [36]. In each round, clients first agree on a pair of public key and private key. Subsequently, each client encrypts their own model updates with the agreed public key and sends the ciphertext to the server. The server then sums the ciphertexts and dispatches the results back to all clients. After that, each client decrypts the results locally with the private key and obtains the corresponding aggregated model updates. Throughout the entire aggregation process, the intermediate results are protected by AHE and cannot be learned by the server.

C. Motivation

The additive cryptographic methods provide robust security guarantees, but still incur significant overhead of computation and communication, making them impractical in use.

Overhead of AHE. AHE encounters significant computation and communication overhead. The computation overhead is mainly caused by complex modular operations performed during the encryption and decryption procedures [16], [37]. We evaluate the average time per round of two tasks with AHE on both client side and server side (see Section V for more details). Figure 1(a) shows that the encryption and the decryption procedure of AHE takes up most of the computation time on the client side, while Figure 1(b) shows that the server spent most of the time (i.e., idle time) waiting for clients to finish computation, which takes up most of the average time per round. Thus, the computation of AHE on clients becomes the bottleneck of the training procedure. Also, the communication overhead of AHE is mainly attributed to the *data inflation* problem [16]. For instance, the Paillier Homomorphic Encryption requires the use of a 2048-bit public key for security, resulting in a large increase in message size. When encrypting a gradient of 32 bits with a 2048-bit public key, the message size inflates by a factor of 64. Given that the communication often occurs over slow network connections with limited bandwidth [21], [22], the data inflation problem further exacerbates the communication.

Overhead of SA. SA also faces the communication overhead mainly due to data inflation. To avoid overflow in the aggregation procedure, SA reserves additional bandwidth, which results in the data inflation issue similar to AHE [17].

D. Compression of Model Updates

As the computation and communication overhead of additive cryptographic methods is proportional to the size of model updates, reducing the size of model updates can alleviate such overhead. Several methods such as traditional gradient compression [18]–[22] and sketch-based gradient compression [22], [38]–[40] have been applied to reduce the size of model updates in distributed machine learning and FL.

Traditional gradient compression. We classify traditional gradient compression methods into two categories, namely *gradient sparsification* and *gradient quantization*. Gradient sparsification approaches [19], [20], [23], [41] reduce the size of model updates by reserving only part of the model updates. And gradient quantization approaches [16], [25], [42] conduct lossy compression by reducing the precision of each element. However, both gradient sparsification and gradient quantization methods have some limitations. First, many existing gradient compression methods lack of additivity, which may lead to erroneous aggregated results when summing the ciphertexts of compressed model updates [19], [20], [23], [25], [41], [43]. Second, both methods struggle for the balance between the reduction of model size and the model accuracy. To reduce the model size, some existing gradient sparsification methods [18], [19], [24], [44] introduce biased model update estimations with biased operators like top-k [24], which leads to a significant model accuracy drop. Also, to maintain the model accuracy, gradient quantization methods [16], [25], [42] only achieve a limited compression ratio of no more than $\times 8$.

	Requirements			
	A	**I**	**L**	**U**
SA	✔	✔		
AHE	✔	✔		
sketch-based methods	✔	✘	✔	✘
SRHT Sketch	✔	✘	✔	✔

Table I: Requirements of additivity (A), integer (I), large compression ratios (L), and unbiasedness (U).

Sketch-based methods. Recent studies attempt to use *sketch* to assist gradient compression [22], [38]–[40]. Sketch belongs to a family of randomized data structures that compress large datasets using compact arrays and random hash functions [45], [46]. Sketch is associated with three operations, namely *compression*, *decompression*, and *aggregation*. Many sketches are naturally additive and support a large compression ratio with bounded error [27], [45], [46]. Among these sketches, the *Subsampled Randomized Hadamard Transform* (SRHT) Sketch [27], [28] is a widely used approach in traditional machine learning. However, existing sketch-based methods still have several limitations. As shown in Table I, existing sketch-based methods adopt sketch to assist quantization or sparsification, which does not meet the unbiasedness requirement or the integer requirement [22], [38], [39]. While it is intuitive to directly adopt SRHT Sketch for compression because of its unbiasedness, it still fails to meet the integer requirement.

III. Overview

We first present the design goals and challenges of FedSSA (Section III-A). We then propose our solutions (Section III-B) and introduce the architecture of FedSSA (Section III-C).

A. Design Goals and Challenges

FedSSA is a compression framework of FL that supports additive cryptographic methods with the following design goals:

- **Generality (G1).** FedSSA is expected to directly support additive cryptographic methods for privacy protection.
- **Efficiency (G2).** FedSSA is expected to significantly reduce the overhead of additive cryptographic methods.

- **Accuracy (G3).** FEDSSA is expected to retain a high model accuracy.

Specifically, FEDSSA is built based on SRHT Sketch (see Section II-D). SRHT Sketch can compress model updates with large compression ratios, therefore reducing the model size and the associated overhead (G2). Also, SRHT Sketch is unbiased, which can retain a high model accuracy (G3). However, SRHT Sketch does not meet the integer requirement. To make SRHT Sketch compatible with additive cryptographic methods, the main challenge is to maintain high model accuracy in two different aspects.

Challenge 1 (C1): Error-bounded integer conversion. Recall in Section II-B that additive cryptographic methods require the plaintext to be in integer form. However, SRHT Sketch takes fractional model updates as input and generates fractional elements. This discrepancy poses a challenge since fractional elements cannot be encrypted by additive cryptographic methods. While taking operations like floor or ceiling function seems to be straightforward solution, these operations are biased and compromise the unbiasedness of SRHT Sketch. Moreover, they may introduce unbounded errors due to precision loss from simply discarding fractional parts. Hence, establishing a theoretical bound to quantify the introduced error is crucial.

Challenge 2 (C2): Efficient refreshing to generate independent hash functions. Recall in Section II-D that SRHT Sketch compresses model updates with hash functions. The independence of hash functions across rounds is crucial when they are treated as random variables. Specifically, this independence ensures the unbiasedness of SRHT Sketch [27]. When using the same hash functions throughout the training procedure, it breaks the independence of hash functions and leads to a notable decline in model accuracy. Refreshing the hash functions of SRHT Sketch is an intuitive idea to maintain independence. However, it is challenging to refresh the hash functions efficiently and correctly. The refreshed hash functions should be synchronized across the server and participating clients. Clients can actively send requests to the server for latest hash functions, yet this approach is inefficient and can burden the server with additional processing overhead. Also, it is crucial to update hash functions correctly on the client side. If the hash function is updated before the client decompresses with the old ones, it leads to incorrect model updates and introduces errors in the FL procedure.

B. Our Solution

To address the aforementioned challenges, FEDSSA is designed based on the following two techniques:

QSRHT Sketch. To address C1, we propose a new sketch called QSRHT Sketch. QSRHT Sketch enhances SRHT Sketch [27] with a *quantization step*. Specifically, it scales the fractional elements to preserve precision of QSRHT Sketch with a parameter called the *quantization scaling factor*. To meet the theoretical guarantee of unbiased estimation, it adopts *probabilistic quantization* to map each fractional element into an integer. The overall introduced error of QSRHT Sketch can be bounded with a theoretical guarantee. Also, we prove that QSRHT Sketch maintains the additivity in expectation, even though the probabilistic quantization is a non-linear operation.

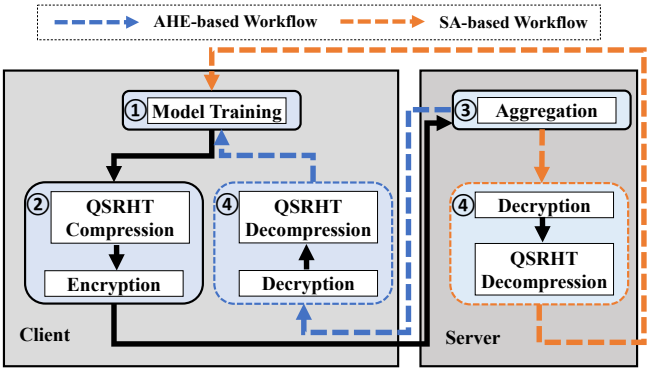


Figure 2: FEDSSA workflow.

Periodic rehashing. To tackle C2, we present a new protocol called the *periodic rehashing*, which refreshes the hash functions in QSRHT Sketch every round to maintain the independence of hash functions. Specifically, the server sends the latest hash functions *along* with the latest model parameters back to clients. This can avoid the requests from clients for new hash functions. After receiving the latest hash functions, the client delays the hash function update procedure until it correctly decompresses with old hash functions, which ensures the correctness of hash function usage.

C. Architecture

FEDSSA includes two parties, namely the *server* and multiple *clients*. FEDSSA supports two workflows, namely *SA-based workflow* and *AHE-based workflow*. SA-based workflow supports secure aggregation while AHE-based workflow supports additive homomorphic encryption.

FEDSSA workflow. As shown in Figure 2, the workflow of FEDSSA consists of four steps:

- **Step (1): Model training.** At the beginning of each round, the server randomly selects a subset of available clients to participate. These clients then acquire the global model and train the model for multiple iterations using the local dataset. In the SA-based workflow, clients fetch the global model from the server, whereas in the AHE-based workflow, clients utilize the locally stored global model.
- **Step (2): Client processing.** The client first updates the hash functions of QSRHT Sketch. Then, the client compresses local model updates with QSRHT Sketch and encrypts QSRHT Sketch using the corresponding cryptographic method.
- **Step (3): Server aggregation.** The server aggregates the encrypted QSRHT Sketches from participating clients by summing them together to obtain the aggregated results.
- **Step (4): Post-processing.** In the SA-based workflow, the post-processing step occurs on the server side, while in the AHE-based workflow, each client executes this step individually. In this step, the corresponding party first decrypts the results obtained in Step (3) to generate the aggregated QSRHT Sketch. Then the party decompresses the aggregated QSRHT Sketch to obtain an unbiased estimation of

Notation	Meaning
	Defined in Section IV-A
S	counter array in QSRHT Sketch
m	number of columns in S
D	sign hash function $\{0, 1, ..., d-1\} \to \{-1, 1\}$
R	index function $\{0, 1, ..., m-1\} \to \{0, 1, ..., d-1\}$
$\mathbf{g}$	original model update vector before compression
$\mathbf{g}'$	model update vector after decompression
d	size of the model update vector $\mathbf{g}$
i	the i-th element in a vector of size d $(0 \leq i \leq d-1)$
j	the j-th element in a vector of size m $(0 \leq j \leq m-1)$
$\mathbf{H}$	normalized Walsh-Hadamard matrix
α	quantization scaling factor
Q	quantization function
r	compression ratio $r = d/m$
K	number of selected clients
	Defined in Section IV-B
t	t-th round in communication
D_t	sign hash function for round t
R_t	index function for round t
	Defined in Section IV-C
$\mathcal{Q}(\cdot)$	probabilistic quantization function
$\mathbf{H}$	d-th normalized Walsh-Hadamard matrix
$\mathbf{D}$	matrix of sign hash functions
$\mathbf{R}$	matrix of index functions
$Comp(\cdot)$	compression operation
$Decomp(\cdot)$	decompression operation
$\mathcal{C}_\alpha(\mathbf{g})$	combination of $Comp(\cdot)$ and $Decomp(\cdot)$
$\mathbb{E}[\cdot]$	expectation
	Defined in Section IV-D
$\mathbf{w}$	model parameters
E	number of iterations for local training
$F(\mathbf{w})$	loss function with $\mathbf{w}$
$\mathbf{g}_{t,v}^u$	the v-th gradient computed by client u in round t

Table II: Major notation used in Section IV.

aggregated model updates. The estimated model updates are then used to refine the global model.

IV. Design

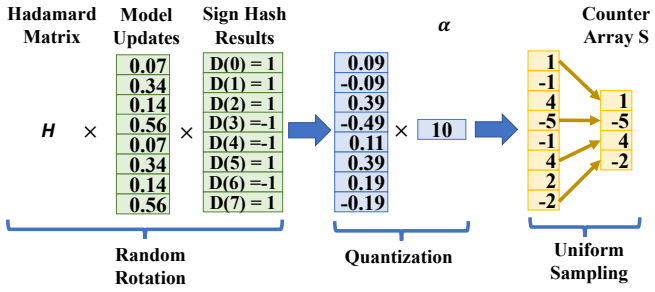
We first introduce QSRHT Sketch (Section IV-A) and the periodic rehashing (Section IV-B). Then we analyze the properties of QSRHT Sketch (Section IV-C) and provide a theoretical guarantee on the convergence of FedSSA (Section IV-D). Table II summarizes the major notations in Section IV.

A. QSRHT Sketch

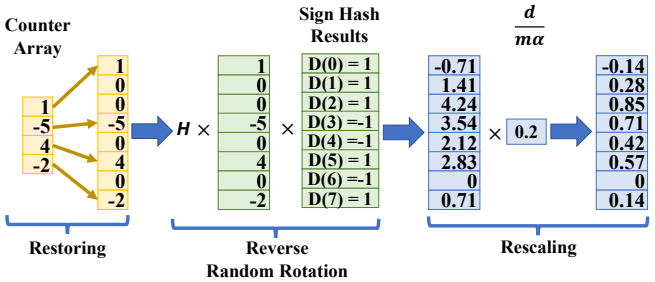
We first introduce the data structure of QSRHT Sketch. Then we introduce the corresponding operations of QSRHT Sketch, namely *compression*, *decompression* and *aggregation*.

Data structure. A QSRHT Sketch comprises a counter array (denoted by S) with m columns and two random hash functions, including the sign hash function (denoted by D) and the index hash function (denoted by R). Let $\mathbf{g}$ be a one-dimensional vector of model updates with d elements, the *compression ratio* of QSRHT Sketch is defined by $r = d/m$. QSRHT Sketch compresses the model updates $\mathbf{g}$ into S via D and R. The function D maps the sign of each element in $\mathbf{g}$ to ± 1 uniformly at random based on the index of the element, i.e., $\{0, 1, ..., d-1\} \to \{-1, 1\}$. The function R uniformly samples m elements from $\mathbf{g}$ with replacements, i.e., $\{0, 1, ..., m-1\} \to \{0, 1, ..., d-1\}$.

Compression. Figure 3(a) shows the compression procedure of QSRHT Sketch with $d = 8$ and $m = 4$, which compresses $\mathbf{g}$ into S by three steps.



(a) Compression of QSRHT Sketch.



(b) Decompression of QSRHT Sketch.

Figure 3: The compression and decompression of QSRHT Sketch.

- **Random rotation.** QSRHT Sketch first randomly flips the sign of each element in $\mathbf{g}$ and gets an intermediate vector $\mathbf{g}_r = (D(0) \times g_0, ...D(i) \times g_i, ..., D(d-1) \times g_{d-1})$, where g_i denotes the i-th element in $\mathbf{g}$ and $0 \leq i \leq d-1$. It then multiplies $\mathbf{g}_r$ by the normalized Walsh-Hadamard matrix [27] $\mathbf{H}$ and obtains the rotated vector $\mathbf{h} = \mathbf{H}\mathbf{g}_r$. The matrix $\mathbf{H}$ is defined as $\mathbf{H} = \frac{1}{\sqrt{d}}\mathbf{H}_d$, where $\mathbf{H}_d$ is computed recursively via $\mathbf{H}_d = \begin{pmatrix} \mathbf{H}_{\frac{d}{2}} & \mathbf{H}_{\frac{d}{2}} \\ \mathbf{H}_{\frac{d}{2}} & -\mathbf{H}_{\frac{d}{2}} \end{pmatrix}$ from an initial matrix $\mathbf{H}_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.
- **Quantization.** To preserve the precision of $\mathbf{h}$, QSRHT Sketch first scales $\mathbf{h}$ by multiplying the quantization scaling factor (defined in Section III-B) (denoted by α) and obtains a scaled vector $\alpha\mathbf{h}$. QSRHT Sketch then applies the probabilistic quantization on the scaled vector $\alpha\mathbf{h}$ element-wisely to obtain the quantized result $\mathbf{h}_q$. Specifically, let Q be the quantization function and h_i be the i-th element in $\mathbf{h}$. The i-th element in $\mathbf{h}_q$ is computed by $Q(\alpha h_i)$, where

$$Q(x) = \begin{cases} \lfloor x \rfloor + 1 & \text{with the probability } p(x) = x - \lfloor x \rfloor, \\ \lfloor x \rfloor & \text{with the probability } 1 - p(x), \end{cases}$$

  and $\lfloor x \rfloor$ denotes the integer no more than an element x. Thus, each element in $\mathbf{h}_q$ is an integer and $\mathbf{h}_q$ is an unbiased estimation of the vector $\alpha\mathbf{h}$.
- **Uniform sampling.** To reduce the size of model updates (i.e., d), QSRHT Sketch performs uniform sampling with the index hash function R to choose m element from $\mathbf{h}_q$ and stores the sampled elements into S. Specifically, for the j-th element in S, $R(j)$ outputs the index of the selected element in $\mathbf{h}_q$. Note that all elements in S are now integers.

Decompression. The decompression procedure is to restore the original model update vector $\mathbf{g}$ from the counter array S.

Figure 3(b) shows the corresponding decompression procedure of QSRHT Sketch, which consists of the following three steps:

- **Restoring.** QSRHT Sketch first restores the counter array of m columns back to an intermediate vector (denoted by $\mathbf{h}'$) with d elements. The i-th element in $\mathbf{h}'$ is computed by $h'_i = \sum_{R(j)=i} S[j]$.
- **Reverse random rotation.** This step can be viewed as the reverse operation of the random rotation step. In this step, QSRHT Sketch first multiplies the corresponding sign hash function to obtain the intermediate result $\mathbf{g}'_r = (D(0) \times h'_0, ...D(i) \times h'_i, ...D(d) \times h'_d)$. Then it multiplies the matrix $\mathbf{H}$ with $\mathbf{g}'_r$ to obtain $\mathbf{g}_h = \mathbf{H}\mathbf{g}'_r$.
- **Rescaling.** Finally, QSRHT Sketch rescales $\mathbf{g}_h$ to obtain the unbiased estimation of the original vector $\mathbf{g}' = \frac{d}{m\alpha}\mathbf{g}_h$, which is the final result of the decompression operation.

Aggregation. The aggregation operation of QSRHT Sketch is used by the server to aggregate QSRHT Sketches from clients. When two QSRHT Sketches have the same parameters (i.e. d, m, and α), they can be aggregated by summing the corresponding values in the counter arrays. While the compression of QSRHT Sketch involves probabilistic quantization, which is non-linear, QSRHT Sketch is still additive in expectation (see Theorem 2 in Section IV-C).

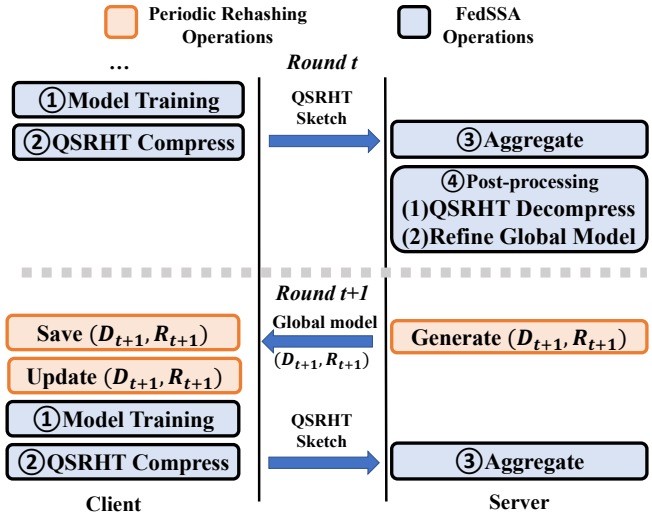
Selection of quantization scaling factor α. The quantization scaling factor α determines the precision preserved in model updates. A larger α preserves more precision for the floating-point values. However, an excessively large α may lead to overflow issues in the scaled vector ($\alpha\mathbf{h}$) which may introduce errors. Denote the number of participating clients by K. To determine an appropriate α, we select α such that $\alpha > \sqrt{\frac{dr}{K}}$ at the beginning of training and use the same value during training We provide the mathematical analysis of α selection in Section IV-D.

B. Periodic Rehashing

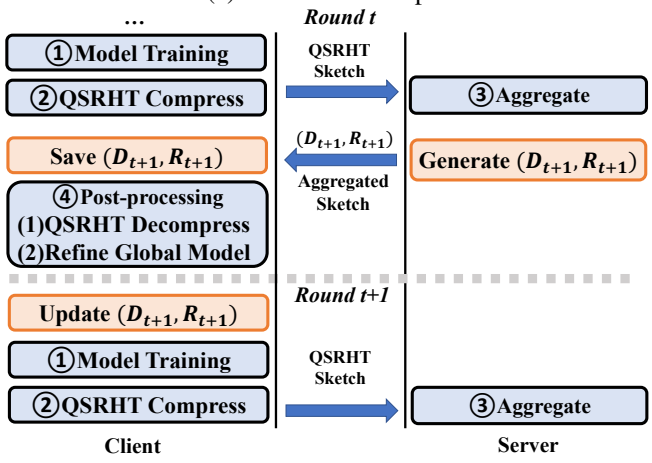
Protocol. The periodic rehashing is designed to refresh the sign hash functions and the index hash function per round. In each round, the server and clients execute the following steps.

- **Server generation.** When sending the latest global model (or aggregated QSRHT Sketch) to clients, the server independently and uniformly at random samples the hash functions D and R. For each message sent to participating clients that contains the global model (or aggregated QSRHT Sketch), the server piggybacks D and R.
- **Client saving.** When each client receives the latest global model (or aggregated QSRHT Sketch) from the server, it also receives the latest hash functions D and R. Instead of directly updating, the client saves D and R.
- **Client updating.** When each client begins its model training, it updates QSRHT Sketch with D and R.

Benefits. The server generation can efficiently synchronize hash functions among clients. The client saving and the client updating can make periodic rehashing compatible with two different workflows and ensure the correctness of hash functions usage (see Section III-C). In the SA-based workflow, the client saving is followed by the client updating, which



(a) SA-based example



(b) AHE-based example

Figure 4: Workflow of periodic rehashing. Encryption and decryption are omitted for simplicity.

is equivalent to an immediate update. However, in the AHE-based workflow, the client should decompress the aggregated QSRHT Sketch with old hash functions. If the client updates hash functions immediately, the corresponding decompression operation would use the new hash functions and results in wrong model updates.

Workflows. Let D_t and R_t denote the sign hash function and index hash function at round t. We show the SA-based and AHE-based workflow of periodic rehashing.

- **SA-based workflow.** Figure 4(a) illustrates the SA-based workflow. In round t, the client trains the model, compresses the updates, and sends them to the server. The server aggregates the QSRHT Sketch and decompresses the updates to refine the global model. Note that in round t, both sides use hash functions (D_t, R_t) for compression and decompression. In round $t+1$, the server sends the updated global model and new hash functions (D_{t+1}, R_{t+1}) to the client, which then updates its hash functions accordingly.
- **AHE-based workflow.** Figure 4(b) shows the AHE-based workflow. In round t, the client sends its QSRHT Sketch to the server. The server aggregates and sends the aggregated

results along with new hash functions (D_{t+1}, R_{t+1}) to each client. The client saves the new hash functions, decompresses the updates using the old hash functions (D_t, R_t) to refine the global model. At the beginning of round $t+1$, the client updates its hash functions before training local model.

Periodic rehashing overhead. Although the periodic rehashing calculates the hash function results in the compression and decompression of QSRHT Sketch on both the server side and clients' side per round, this process incurs only a small computation overhead in FL. We evaluate the overhead of periodic rehashing and show that the overhead of periodic rehashing is acceptable in FL (see Exp#6 in Section V-B).

C. Analysis of QSRHT Sketch

We analyze the theoretical guarantees of QSRHT Sketch, including unbiasedness, additivity and bounded variance property. For a QSRHT Sketch with the quantization scaling factor $\alpha \in \mathbb{R}$, we denote the compression procedure by $Comp(\cdot)$ and the corresponding decompression procedure by $Decomp(\cdot)$. For a model update vector $\mathbf{g} \in \mathbb{R}^d$, we denote the combination of compression and decompression procedure by $\mathcal{C}_\alpha(\mathbf{g}) = Decomp(Comp(\mathbf{g}))$.

Theoretical guarantees. Theorem 1 provides the unbiasedness of QSRHT Sketch with periodic rehashing. Theorem 2 provides the additivity guarantee of QSRHT Sketch, which enables the aggregation operation of QSRHT Sketch. Theorem 3 offers a theoretical guarantee on the variance of model update estimations produced by QSRHT Sketch. In the interest of space, we put the proofs of Theorem 1–Theorem 3 to Appendix B.

Theorem 1. *Given $\alpha \in \mathbb{R}$ and $\mathbf{g} \in \mathbb{R}^d$, QSRHT Sketch is an unbiased estimator of $\mathbf{g}$, i.e. $\mathbb{E}[\mathcal{C}_\alpha(\mathbf{g})] = \mathbf{g}$. The expectation is over the sign hash function and the index hash function used in the corresponding QSRHT Sketch.*

Theorem 2. *Given $\alpha \in \mathbb{R}$, for any $\mathbf{g_1}, \mathbf{g_2} \in \mathbb{R}^d$, QSRHT Sketch satisfies $\mathbb{E}[\mathcal{C}_\alpha(\mathbf{g_1} + \mathbf{g_2})] = \mathbb{E}[\mathcal{C}_\alpha(\mathbf{g_1}) + \mathcal{C}_\alpha(\mathbf{g_2})]$.*

Theorem 3. *Assume that quantization scaling factor $\alpha \in \mathbb{R}$ is given. For any model update vector $\mathbf{g} \in \mathbb{R}^d$, we have*

$$\mathbb{E}[\|\mathcal{C}_\alpha(\mathbf{g}) - \mathbf{g}\|_2^2] = \frac{(d+m-1)d}{4m\alpha^2} + \frac{d-1}{m}\|\mathbf{g}\|_2^2. \quad (1)$$

The necessity of periodic rehashing. Note that Theorem 1 and the periodic rehashing together establish the unbiasedness of QSRHT Sketch in FEDSSA. Theorem 1 relies on the randomness of hash functions to guarantee the unbiasedness of QSRHT Sketch. However, if the same hash functions are used throughout the training procedure, the unbiasedness of QSRHT Sketch no longer holds.

Variance analysis of QSRHT Sketch. Theorem 3 establishes a theoretical bound for the variance of the model update estimation. The term $\frac{d-1}{m}||g||_2^2$ can be viewed as the *sampling variance*. The coefficient of the sampling variance $\frac{d-1}{m} \approx \frac{d}{m}$ represents the compression ratio of QSRHT Sketch, which captures the trade-off between compression ratio and variance.

D. Convergence Analysis of FEDSSA

Notations. We first introduce some notations. The FL optimization problem involves N clients. The goal is to learn the model parameter (denoted by $\mathbf{w}$) collaboratively. The federated optimization problem is defined as $\min_w F(w) = \frac{1}{N}\sum_{i=1}^N F_i(\mathbf{w})$, where $F_i(\mathbf{w}) = \mathbb{E}_{x\sim\mathcal{D}_i}[l(\mathbf{w}, x)]$ and $l(\mathbf{w}, x)$ denotes the loss function evaluated on training sample drawn from the local training set $\mathcal{D}_i$ of the i-th client. In each round, FEDSSA server randomly selects K clients without replacement and each client performs E iterations for local training. In each round t, we denote the current model parameter by $\mathbf{w}_t$ and the learning rate by η_t, and we denote the j-th iteration gradient of the client i by $\mathbf{g}_{t,v}^u$.

Assumptions. We make the following assumptions which are standard in FL convergence analysis [47]. Specifically, Assumption 1 assumes Lipschitz continuous gradients on loss functions. Assumption 2 and 3 assume the stochastic gradient of each client to be unbiased and bounded.

Assumption 1 (Smoothness)**.** *There exists a constant $L \in \mathbb{R}$, $L > 0$, such that $\|\nabla F_i(\mathbf{u}) - \nabla F_i(\mathbf{v})\| \leq L\|\mathbf{u} - \mathbf{v}\|$, $\forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^d$, $\forall 1 \leq i \leq N$.*

Assumption 2 (Unbiasedness)**.** *The local gradient estimator is unbiased, i.e. $\mathbb{E}\left[\mathbf{g}_{t,v}^u\right] = \nabla F_i(\mathbf{w}_t)$, $\forall 1 \leq i \leq N$, where the expectation is over all possible sample batches.*

Assumption 3 (Bounded variance)**.** *There exist two constants $\sigma_L > 0, \sigma_G > 0$, such that the variance of each local gradient estimator is bounded by $\mathbb{E}\left[\|\mathbf{g}_{t,v}^u - \nabla F_i(\mathbf{w}_t)\|^2\right] \leq \sigma_L^2$, $\forall 1 \leq i \leq N$. The difference between each local loss function F_i and the global loss function F can be bounded by $\|\nabla F_i(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)\|^2 \leq \sigma_G^2$, $\forall 1 \leq i \leq N$, $\forall 1 \leq t \leq T$.*

Convergence analysis. This convergence analysis is based on the analysis in [47]. When assuming that all clients participate in each round (i.e. $N = K$), we can establish the following theorem. Theorem 4 provides a convergence bound for FEDSSA. We leave the proof of the Theorem 4 to Appendix C.

Theorem 4. *Suppose that η is a constant real number such that $\eta \leq \min\left(\frac{1}{8EL}, \frac{1}{EL\left(1+\frac{d-1}{m}\right)}\right)$. When using a constant learning rate $\eta_t = \eta$ for each communication round and for the local training of each client, the following convergence bound holds for* FEDSSA*:*

$$\min_{1\leq t\leq T}\{\mathbb{E}\|\nabla F(\mathbf{w}_t)\|^2\} \leq \frac{F(\mathbf{w}_0) - F(\mathbf{w}_*)}{c\eta ET} + M, \quad (2)$$

where $F(\mathbf{w}_*) \coloneqq \min_{\mathbf{w}}\{F(\mathbf{w})\}$,

$$M \coloneqq \frac{1}{c}\left[\frac{L\eta}{E}c_2 + \frac{L\eta}{2K}\left(1 + \frac{d-1}{m}\right)\sigma_L^2 + \frac{5E\eta^2L^2}{2}c_3\right],$$

where c is a constant, $c_2 \coloneqq \frac{d(1+r)}{4K\alpha^2}$ and $c_3 \coloneqq \sigma_L^2 + 6E\sigma_G^2$.

Quantizaion scale selection. The term $\frac{d(1+r)}{4K\alpha^2}$ in equation (2) can be interpreted as the quantization error of QSRHT Sketch. Theorem 4 suggests that by carefully selecting the quantization scaling factor, the quantization step in QSRHT Sketch has a

minimal impact on the final model accuracy. In the probabilistic quantization step, QSRHT Sketch chooses a quantization scale factor such that $\alpha \geq \left(\sqrt{\frac{d \times r}{K}}\right)$.

V. Evaluation

A. Experiment Settings

Implementation details. To implement the FedSSA prototype, we first develop a single-machine simulator in Python (∼300 LoC) to simulate the training procedure. Then we develop the FedSSA server module (∼1,600 LoC) and the FedSSA client module (∼160 LoC) on top of the simulator with Python. We also develop QSRHT Sketch with PyTorch on GPU (∼180 LoC). For AHE, we adopt the Paillier's scheme [15]. We use Intel's Paillier Cryptosystem Library to implement the encryption and decryption procedure.

Datasets and models. We conduct image classification tasks using the CIFAR-10 and CIFAR-100 datasets. Specifically, we train a ResNet9 [32] model with $d = 6,573,120$ trainable parameters for CIFAR-10 and a ResNet18 [33] model with $d = 11,220,132$ trainable parameters for CIFAR-100. In both cases, we replace batch norm [48] with group norm [49] based on the recommendations from a previous study [50].

Metrics. We use the following two metrics for evaluation, namely the *test accuracy* and *average computation time*. Test accuracy is the percentage of correctly classified samples in the test dataset. The average computation time represents the average time spent on computation by the client throughout the procedure. We use the average computation time to evaluate the computation overhead when AHE is in use [16] (Exp#7).

Data allocation. In our experiments on CIFAR-10 and CIFAR-100, we simulate non-independent and identically distributed (non-i.i.d.) sample distributions among N clients using the Dirichlet distribution $Dir(\beta)$. This approach is commonly used in federated learning research [51]. We consider three values of $\beta = 0.1, 0.3, 0.5$ to represent different levels of non-i.i.d. data distribution, where smaller β implies a higher degree of non-i.i.d. For simplicity, we use CIFAR-10 (β) and CIFAR-100 (β) to refer to the datasets with different β.

FL settings. We evaluate FedSSA under two different settings, namely *SA setting* and *AHE setting*.

- **SA setting.** In this setting, the FL procedure iterates for $T = 1000$ rounds involving $N = 100$ clients. In each round, the FedSSA randomly samples $K = 12$ clients without replacement for participation, The learning rate η_t of each round is determined by a cosine annealing scheduler with initial learning rate $\eta_0 = 0.1$. We use this setting to evaluate the test accuracy of FedSSA and other baselines (Exp#1). We also use the SA setting by default when evaluating the design effectiveness of FedSSA (Exp#3–Exp#6).
- **AHE setting.** In this setting, the FL procedure iterates for $T = 100$ rounds involving $N = 10$ clients. In each round, the FedSSA chooses all clients for participation (i.e. $K = 10$). We use a fixed learning rate $\eta = 0.01$ in the training procedure. As for the Paillier encryption, we set the key size to 2048 bits in our evaluation. We use this setting to evaluate the test accuracy on AHE (Exp#2) and computational overhead reduction on AHE (Exp#7).

Each participating client performs $E = 3$ local rounds with a local batch size of $B = 64$. This means each client iterates over its local dataset for E times, consistent with the approach used in a previous study [4]. For FedSSA, we use a default $\alpha = 10^6$, which satisfies the requirement of quantization scale selection under different settings. We also vary α to further study the impact of α (Exp#4).

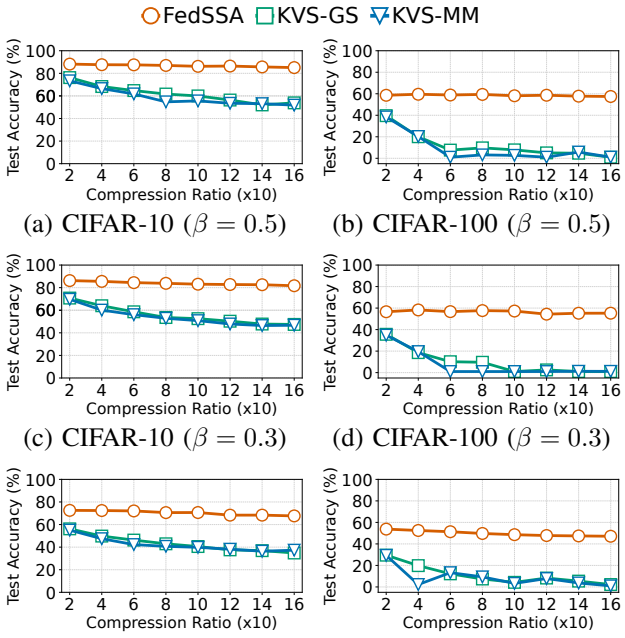
Baseline algorithms. We evaluate the test accuracy of FedSSA with six baseline algorithms from three categories:

- **SA-based compressor.** We compare FedSSA with two SA-based compression algorithms, namely KVSAgg-Minmax (KVS-MM) and KVSAgg-GSpar (KVS-GS) [29]. As they support a large compression ratio, we vary the compression ratio from 20 to 160 with a step size of 20, and we refer to them as *large compression ratios*. To achieve the actual compression ratio of r and ensure correct decompression, we configure KVS-MM and KVS-GS with specific parameters $W = \frac{d}{11.25 \times K \times r}$ and $m = \frac{d}{9 \times r}$ as suggested in [29].
- **AHE-based compressor.** We compare FedSSA with one AHE-based algorithm, namely BatchCrypt (BC) [16]. BC is a quantization-based method and only supports a small compression ratio (i.e., 2 and 4). We set the compression ratio as 2 and 4 for their comparison and refer to them as *small compression ratios*, while we also validate the test accuracy of FedSSA under large compression ratios.
- **Non privacy-preserving compressor.** To show that FedSSA can retain test accuracy while achieving a large compression ratio, we further compare FedSSA with three baseline gradient compression algorithms without privacy protection in the SA setting. We choose two sparsification methods and one sketch-based method as baselines. Specifically, for the gradient sparsification methods, we choose Minmax sampling (MM) [30], GSpar sampling (GS) [31] and random-k sampling (RK) [52] with shared randomness (i.e. choosing the same elements among clients). For the sketch-based method, we choose FetchSGD (FSGD) [22] as baseline.

B. Experiment Results

(Exp#1) Accuracy comparison with SA-based compressors. We compare the test accuracy of FedSSA with the two SA-based compressors under the SA setting. We conduct the experiment on six different training tasks: CIFAR-10 ($\beta = 0.5, 0.3, 0.1$) and CIFAR-100 ($\beta = 0.5, 0.3, 0.1$).

Figure 5 shows that FedSSA consistently outperforms KVS-MM and KVS-GS in the test accuracy across different training tasks by up to 34.25% with varying the compression ratio. Moreover, FedSSA experiences less test accuracy degradation as the compression ratio increases. Figures 5 (a), 5(c), and 5(e) show that FedSSA achieves the test accuracy of 88.24%, 86.24%, and 72.56% respectively when $r = 20$. Upon increasing r to 160, FedSSA experiences only a slight accuracy reduction of 3.10%, 4.50%, and 4.91% respectively.

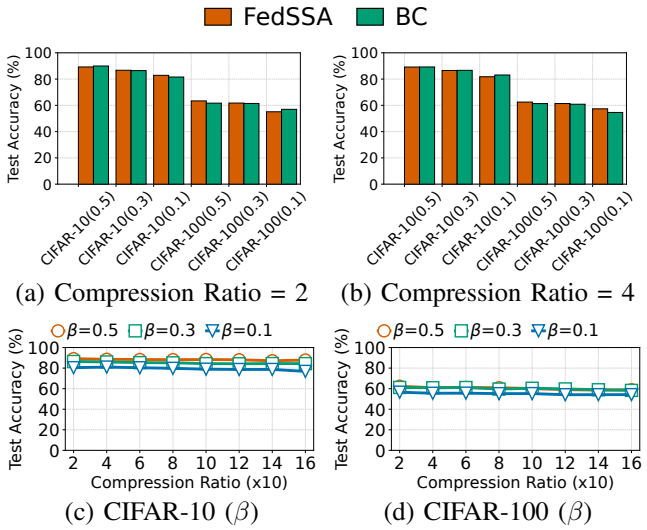


(a) CIFAR-10 ($\beta = 0.5$) (b) CIFAR-100 ($\beta = 0.5$)

(c) CIFAR-10 ($\beta = 0.3$) (d) CIFAR-100 ($\beta = 0.3$)

(e) CIFAR-10 ($\beta = 0.1$) (f) CIFAR-100 ($\beta = 0.1$)

Figure 5: (Exp#1) Accuracy comparison with secure aggregation-based compressors.



(a) Compression Ratio = 2 (b) Compression Ratio = 4

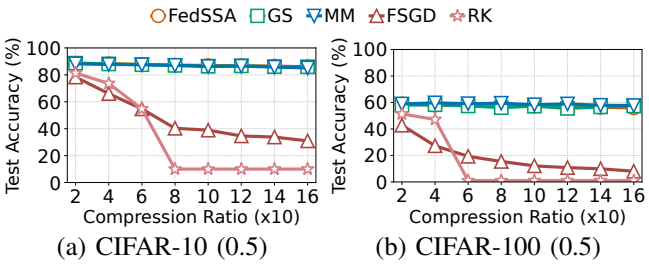
(c) CIFAR-10 (β) (d) CIFAR-100 (β)

Figure 6: (Exp#2) Accuracy comparison with AHE-based compressor.

In contrast, both KVS-MM and KVS-GS suffer from significant accuracy degradation of more than 17.76% when the compression ratio increases to $r = 160$.

(Exp#2) Accuracy comparison with AHE-based compressor. We first compare the test accuracy of FEDSSA with BC under the AHE setting on CIFAR-10 ($\beta = 0.5, 0.3, 0.1$) and CIFAR-100 ($\beta = 0.5, 0.3, 0.1$). Figures 6(a) and 6(b) show that FEDSSA achieves a comparable test accuracy with BC under small compression ratios (2 to 4) across different datasets. The absolute differences between FEDSSA and BC are within 1.86% and 1.33% across different datasets for compression ratios 2 and 4, respectively.

We next validate the test accuracy of FEDSSA for large compression ratios under the AHE setting. Figures 6(c) and 6(d) show that the FEDSSA achieves a stable model accuracy with no more than 4.19% and 3.8% of accuracy drop on CIFAR-10 and CIFAR-100, respectively, with the compression ratio ranging from 20 to 160. Also, we compare the test accuracy of FEDSSA with the compression ratio of 20 to that of BC with a compression ratio of 4. For CIFAR-10, the test accuracy of FEDSSA is lower than that of BC within 2.2%, while for CIFAR-100, the test accuracy of FEDSSA is higher than that of BC by up to 2.76%. For CIFAR-10, the test accuracy of FEDSSA is within 2.2% lower than that of BC. This shows the potential of FEDSSA in preserving model accuracy, particularly on large neural networks.



(a) CIFAR-10 (0.5) (b) CIFAR-100 (0.5)

Figure 7: (Exp#3) Accuracy comparison with non-privacy compressors.

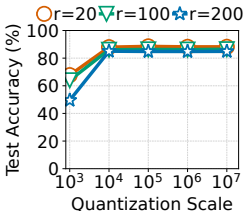


Figure 8: (Exp#4) Impact of α.

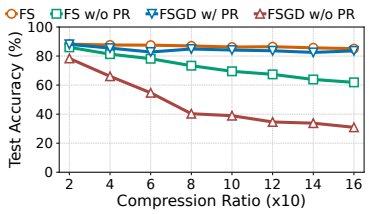


Figure 9: (Exp#5) Effect of periodic rehashing.

(Exp#3) Accuracy comparison with non-privacy compressors. We evaluate the test accuracy of FEDSSA and the 4 baseline compressors without privacy protection. We fix $\beta = 0.5$ for two training tasks: CIFAR-10 (0.5) and CIFAR-100 (0.5).

Figure 7 illustrates that FEDSSA achieves comparable test accuracy compared to baselines on both training tasks. FEDSSA retains higher model accuracy compared to FS due to its unbiasedness. Similarly, FEDSSA outperforms RK due to the bounded-error guarantee provided by QSRHT Sketch. For the CIFAR-10 ($\beta = 0.5$) task, MM, GS, FSGD, RK, and FEDSSA achieve test accuracies of 88.7%, 88.53%, 78.35%, 81.31% and 88.24%, respectively for $r = 20$. As the compression ratio increases, the test accuracy decreases by 2.6%, 2.7%, 47.42%, and 3.10% for MM, GS, FSGD, and FEDSSA, respectively. Meanwhile, the test accuracy of RK rapidly decreases to 1% when r = 60. For the CIFAR-100 ($\beta = 0.5$) task, MM, GS, and FEDSSA maintain stable test accuracies ranging from 59.69% to 55.86%. In contrast, the test accuracy of FSGD and RK achieves only 42.82% and 51.33% when $r = 20$ and quickly drops to 12.15% and 1% when $r = 120$.

(Exp#4) Impact of quantization scaling factor. We evaluate

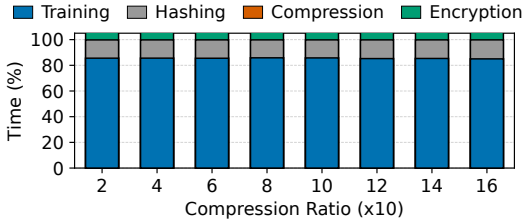


Figure 10: (Exp#6) FEDSSA overhead.

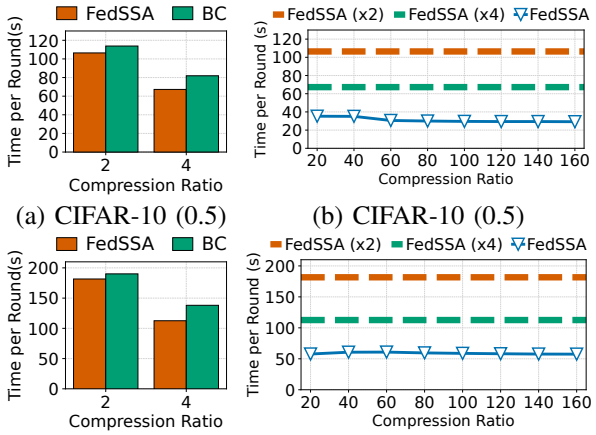
the test accuracy by varying the quantization scaling factor (i.e., α) on the CIFAR-10 (0.5) training task. We consider three different compression ratios: $r = 20, 100, 200$. For each compression ratio group, we vary α and analyze its impact on the test accuracy of FEDSSA. Furthermore, we evaluate the final test accuracy of FEDSSA without the quantization step to understand the overall impact of quantization.

Figure 8 shows that for the CIFAR-10 (0.5) task, when $\alpha = 10^3$, the test accuracy of FEDSSA with quantization drops to 67.71%, 63.97%, and 49.53%, respectively. However, when setting $\alpha \geq 10^4$, the test accuracy of FEDSSA without quantization reaches 87.96%, 86.41%, and 84.60% for different compression ratios ($r = 20, 100, 200$), respectively. When using an even larger α (e.g., 10^5, 10^6, and 10^7), the differences in the test accuracy are insignificant, i.e., within 0.4%.

(Exp#5) Effect of periodic rehashing. We evaluate the impact of periodic rehashing on the final test accuracy on the CIFAR-10 (0.5) training task under the SA setting. We categorize the compressors into two groups: (i) The first group includes FEDSSA with periodic rehashing (FS) and FEDSSA without periodic rehashing (FS w/o PR). (ii) The second group consists of FetchSGD with periodic rehashing (FSGD w/ PR) and FetchSGD without periodic rehashing (FSGD w/o PR). To integrate periodic rehashing into FetchSGD, we modify the error feedback step as the last step of the decompression operation [22]. For each group, we vary the compression ratio and evaluate the final test accuracy.

Figure 9 shows that across different groups, the sketch compressor with periodic rehashing consistently outperforms the non-rehashed sketch compressor in test accuracy. The test accuracy of FEDSSA with periodic rehashing varies only from 88.24% to 85.08%, with a minor decrease of 3.16%. In contrast, FEDSSA without periodic rehashing exhibits a significant accuracy drop, ranging from 86.01% to 61.89%, with a decrease of over 24%. Similarly, in the second group, the final test accuracy of FetchSGD with periodic rehashing ranges from 88.08% to 83.58%, with an accuracy loss of 4.5%. On the other hand, FetchSGD without periodic rehashing experiences a substantial test accuracy drop, ranging from 78.35% to 30.93%. These results highlight the importance of periodic rehashing in maintaining higher test accuracy.

(Exp#6) FEDSSA overhead. In the SA setting, we assess the computation overhead of FEDSSA on the client's side for CIFAR-10 (0.5). Specifically, we analyze the percentage of local training (Training), compression with QSRHT Sketch, and encryption (Encryption) with SA under various compression ratios. We break down the computation time of compression into hash function calculation (Hashing) and other operations except for hash function calculation (Compression).



(a) CIFAR-10 (0.5) (b) CIFAR-10 (0.5)

(c) CIFAR-100 (0.5) (d) CIFAR-100 (0.5)

Figure 11: (Exp#7) AHE computational overhead reduction.

Figure 10 shows that the periodic rehashing computation overhead of compression (excluding hash calculation) and encryption is negligible across different compression ratios. The majority of computation time is consumed by local training and hash calculation, which takes up around 85% and around 14.8%, respectively. The hash calculation is insignificant when compared to local training. Therefore, the overhead of periodic rehashing is acceptable in the evaluation.

(Exp#7) AHE computational overhead reduction. To show that FEDSSA can reduce the computational overhead with a larger compression ratio, we compare the average computation time of FEDSSA and BC under low compression ratios. We also evaluate the average computation time of FEDSSA under large compression ratios. We focus on CIFAR-10 (0.5) and CIFAR-100(0.5).

Figures 11(a) and 11(c) show that FEDSSA takes less computation time than BC under small compression ratios. For the compression ratio of 4, the average computation time of FEDSSA is less than that of BC by 17.9% and 18.49% on CIFAR-10 (0.5) and CIFAR-100 (0.5), respectively. FEDSSA takes less average computation time with larger compression ratios. Figures 11(b) and 11(d) show that the average computation time of FEDSSA is 35.25 seconds and 57.71 seconds on CIFAR-10 (0.5) and CIFAR-100 (0.5), respectively, with the compression ratio of 20. This represents a reduction of 56.71% and 58.15%, respectively compared to BC with a compression ratio of 4. Meanwhile, the accuracy drop is no more than 2.2%, which is acceptable (see Exp#2).

VI. RELATED WORK

Gradient quantization. These methods reduce the gradient size by reducing the precision of each gradient element [16], [25], [42], [53], [54]. However, these quantization methods are originally designed for model updates and maybe not suitable for the quantization procedure of the sketch. In contrast, the quantization step of QSRHT Sketch can provide a bounded-error guarantee while maintaining the additivity of sketch.

Sketch-based gradient compression. Existing sketch-based compression methods mainly focus on two aspects: (i) assisting quantization or sparsification with sketch [22], [38], [55]; (ii) addressing the computation overhead of sketch in distributed machine learning [40]. Unfortunately, these methods inherit the limitations of gradient quantization or gradient sparsification, which is addressed by FEDSSA. The most similar work to ours is SQuaFL [56], which combines quantization with Count Sketch for compression [45]. However, FEDSSA addresses the following issues that SQuaFL does not consider: (i) adopts QSRHT Sketch with provable quantization scale factor which satisfies additivity (instead of lacking additivity); (ii) providing a bounded-error guarantee (instead of potential numerical overflow in the sketch compression procedure).

Differential privacy. Differential privacy (DP) protects client's model updates by introducing additional noise [57]–[59]. Some studies further codesign gradient compression with specific DP mechanisms [39], [60]–[63]. While these studies focus on alleviating the communication overhead of DP-based methods, FEDSSA primarily supports additivity cryptographic methods, which do not compromise model accuracy for privacy protection [10], [16], [64], [65].

VII. CONCLUSION

In this paper, we propose FEDSSA, a compression framework that can reduce the overhead of additive cryptographic methods in FL while retaining model accuracy. FEDSSA propose QSRHT Sketch, which satisfies the integer requirement with theoretical-guaranteed quantization and supports a large compression ratio with bounded error. FEDSSA also proposes periodic rehashing to fresh the hash functions and ensures the correctness of hash function usage. Our evaluation shows that FEDSSA can achieve comparable results to state-of-the-art SA-based compressors and AHE-based compressor under the same compression ratio. Furthermore, when compared to state-of-the-art AHE-based compressors, FEDSSA decreases the computation time per round by up to 58.15% with a model accuracy loss of no more than 2.2%.

REFERENCES

[1] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[2] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *Journal of Healthcare Informatics Research*, vol. 5, pp. 1–19, 2021.

[3] G. Long, Y. Tan, J. Jiang, and C. Zhang, "Federated learning for open banking," in *Federated Learning: Privacy and Incentive*. Springer, 2020, pp. 240–254.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. of AISTAT*, 2017.

[5] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Proc. of NeurIPS*, 2019.

[6] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?" *Proc. of NeurIPS*, 2020.

[7] Y. Huang, S. Gupta, Z. Song, K. Li, and S. Arora, "Evaluating gradient inversion attacks and defenses in federated learning," *Proc. of NeurIPS*, 2021.

[8] D. I. Dimitrov, M. Balunovic, N. Konstantinov, and M. Vechev, "Data leakage in federated averaging," *Trans. on Machine Learning Research*, 2022.

[9] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–36, 2021.

[10] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. of ACM CCS*, 2017.

[11] Y. Aono, T. Hayashi, L. Wang, S. Moriai *et al.*, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2017.

[12] M. Rathee, C. Shen, S. Wagh, and R. A. Popa, "Elsa: Secure aggregation for federated learning with malicious actors," in *Proc. of IEEE S&P*, 2022.

[13] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly) logarithmic overhead," in *Proc. of ACM CCS*, 2020.

[14] J. So, C. He, C.-S. Yang, S. Li, Q. Yu, R. E Ali, B. Guler, and S. Avestimehr, "Lightsecagg: a lightweight and versatile design for secure aggregation in federated learning," *Proc. of MLSys*, 2022.

[15] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International conference on the theory and applications of cryptographic techniques*. Springer, 1999, pp. 223–238.

[16] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning," in *Proc. of USENIX ATC*, 2020.

[17] K. Bonawitz, F. Salehi, J. Konečnỳ, B. McMahan, and M. Gruteser, "Federated learning with autotuned communication-efficient secure aggregation," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2019, pp. 1222–1226.

[18] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns," in *Proc. of INTERSPEECH*, 2014.

[19] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.

[20] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," *Proc. of NeurIPS*, 2017.

[21] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *Proc. of AISTAT*, 2020.

[22] D. Rothchild, A. Panda, E. Ullah, N. Ivkin, I. Stoica, V. Braverman, J. Gonzalez, and R. Arora, "Fetchsgd: Communication-efficient federated learning with sketching," in *Proc. of ICML*, 2020.

[23] S. Agarwal, H. Wang, S. Venkataraman, and D. Papailiopoulos, "On the utility of gradient compression in distributed training systems," *Proc. of MLSys*, 2022.

[24] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," *arXiv preprint arXiv:1704.05021*, 2017.

[25] M. Li, R. B. Basat, S. Vargaftik, C. Lao, K. Xu, X. Tang, M. Mitzenmacher, and M. Yu, "Thc: Accelerating distributed deep learning using tensor homomorphic compression," *arXiv preprint arXiv:2302.08545*, 2023.

[26] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi, "Error feedback fixes signsgd and other gradient compression schemes," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3252–3261.

[27] Y. Lu, P. Dhillon, D. P. Foster, and L. Ungar, "Faster ridge regression via the subsampled randomized hadamard transform," *Proc. of NeurIPS*, 2013.

[28] J. Lacotte, S. Liu, E. Dobriban, and M. Pilanci, "Optimal iterative sketching methods with the subsampled randomized hadamard transform," *Proc. of NeurIPS*, 2020.

[29] Y. Wu, S. Dong, Y. Zhou, Y. Zhao, F. Fu, T. Yang, C. Niu, F. Wu, and B. Cui, "Kvsagg: Secure aggregation of distributed key-value sets," in *Proc. of IEEE ICDE*, 2023.

[30] Y. Zhao, Y. Zhang, Y. Li, Y. Zhou, C. Chen, T. Yang, and B. Cui, "Minmax sampling: A near-optimal global summary for aggregation in the wide area," in *Proc. of ACM SIGMOD*, 2022.

[31] S. Wang, "A practical guide to randomized matrix computations with matlab implementations," *arXiv preprint arXiv:1505.07570*, 2015.

[32] D. Page. (2024) How to train your resnet. [Online]. Available: https://myrtle.ai/how-to-train-your-resnet/

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[34] D. Huba, J. Nguyen, K. Malik, R. Zhu, M. Rabbat, A. Yousefpour, C.-J. Wu, H. Zhan, P. Ustinov, H. Srinivas *et al.*, "Papaya: Practical, private, and scalable federated learning," *Proc. of MLSys*, 2022.

[35] Z. Jiang, W. Wang, and R. Chen, "Taming client dropout for distributed differential privacy in federated learning," *arXiv preprint arXiv:2209.12528*, 2022.

[36] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*. Springer, 2017, pp. 409–437.

[37] W. Jin, Y. Yao, S. Han, C. Joe-Wong, S. Ravi, S. Avestimehr, and C. He, "Fedml-he: An efficient homomorphic-encryption-based privacy-preserving federated learning system," *arXiv preprint arXiv:2303.10837*, 2023.

[38] N. Ivkin, D. Rothchild, E. Ullah, I. Stoica, R. Arora *et al.*, "Communication-efficient distributed sgd with sketching," *Proc. of NeurIPS*, 2019.

[39] W.-N. Chen, C. A. C. Choo, P. Kairouz, and A. T. Suresh, "The fundamental price of secure aggregation in differentially private federated learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 3056–3089.

[40] J. Gui, Y. Song, Z. Wang, C. He, and Q. Huang, "Sk-gradient: Efficient communication for distributed machine learning with data sketch," in *Proc. of IEEE ICDE*, 2023.

[41] L. Zhang, L. Zhang, S. Shi, X. Chu, and B. Li, "Evaluation and optimization of gradient compression for distributed deep learning," *arXiv preprint arXiv:2306.08881*, 2023.

[42] K. Mishchenko, B. Wang, D. Kovalev, and P. Richtárik, "Intsgd: Adaptive floatless compression of stochastic gradients," *arXiv preprint arXiv:2102.08374*, 2021.

[43] S. Vargaftik, R. Ben-Basat, A. Portnoy, G. Mendelson, Y. Ben-Itzhak, and M. Mitzenmacher, "Drive: One-bit distributed mean estimation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 362–377, 2021.

[44] A. Beznosikov, S. Horváth, P. Richtárik, and M. Safaryan, "On biased compression for distributed learning," *Journal of Machine Learning Research*, vol. 24, no. 276, pp. 1–50, 2023.

[45] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," in *Proc. of ICALP*, 2002.

[46] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.

[47] H. Yang, M. Fang, and J. Liu, "Achieving linear speedup with partial worker participation in non-iid federated learning," *arXiv preprint arXiv:2101.11203*, 2021.

[48] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. of ICML*, 2015.

[49] Y. Wu and K. He, "Group normalization," in *Proc. of ECCV*, 2018.

[50] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-iid data quagmire of decentralized machine learning," in *Proc. of ICML*, 2020.

[51] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," in *Proc. of IEEE ICDE*, 2022.

[52] S. Shi, X. Chu, K. C. Cheung, and S. See, "Understanding top-k sparsification in distributed deep learning," *arXiv preprint arXiv:1911.08772*, 2019.

[53] A. T. Suresh, X. Y. Felix, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," in *Proc. of ICML*, 2017.

[54] J. Xin, M. Canini, P. Richtárik, and S. Horváth, "Global-qsgd: Practical floatless quantization for distributed learning with theoretical guarantees," *arXiv preprint arXiv:2305.18627*, 2023.

[55] J. Jiang, F. Fu, T. Yang, and B. Cui, "Sketchml: Accelerating distributed machine learning with data sketches," in *Proc. of ACM SIGMOD*, 2018.

[56] P. Prakash, J. Ding, M. Shu, J. Wang, W. Xu, and M. Pan, "Squafl: Sketch-quantization inspired communication efficient federated learning," in *2021 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2021, pp. 350–354.

[57] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proc. of ACM CCS*, 2016, pp. 308–318.

[58] S. Truex, L. Liu, K.-H. Chow, M. E. Gursoy, and W. Wei, "Ldp-fed: Federated learning with local differential privacy," in *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, 2020, pp. 61–66.

[59] Y. Zhao, J. Zhao, M. Yang, T. Wang, N. Wang, L. Lyu, D. Niyato, and K.-Y. Lam, "Local differential privacy-based federated learning for internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 8836–8853, 2020.

[60] T. Li, Z. Liu, V. Sekar, and V. Smith, "Privacy for free: Communication-efficient learning with differential privacy using sketches," *arXiv preprint arXiv:1911.00972*, 2019.

[61] B. Wang, F. Wu, Y. Long, L. Rimanic, C. Zhang, and B. Li, "Datalens: Scalable privacy preserving training via gradient compression and aggregation," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 2146–2168.

[62] R. Kerkouche, G. Ács, C. Castelluccia, and P. Genevès, "Compression boosts differentially private federated learning," in *Proc. of IEEE (EuroS&P)*, 2021.

[63] Y. Youn, Z. Hu, J. Ziani, and J. Abernethy, "Randomized quantization is all you need for differential privacy in federated learning," *arXiv preprint arXiv:2306.11913*, 2023.

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

[65] Z. Zeng, Y. Du, Z. Fang, L. Chen, S. Pu, G. Chen, H. Wang, and Y. Gao, "Flbooster: A unified and efficient platform for federated learning acceleration," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 3140–3153.

[66] Q. Huang, S. Sheng, X. Chen, Y. Bao, R. Zhang, Y. Xu, and G. Zhang, "Toward {Nearly-Zero-Error} sketching via compressive sensing," in *Proc. of USENIX NSDI*, 2021.

APPENDIX A
CONFIGURATIONS OF EXPERIMENTS

The configurations of Exp#1 We justify the configuration in Exp#1 as follows. For FEDSSA, when the compression ratio is r, the FEDSSA utilizes a QSRHT sketch with 1 row and $\frac{d}{r}$ columns. For KVSAgg-Minmax, two configurations need to be configured, namely the number of columns for the IBLT table m and the number of elements for local sampling W. Specifically, IBLT uses $s = 3$ hash tables, each of which consists of m buckets. Each bucket contains a frequency counter (4 bytes), the key sum counter (4 bytes) and value counter (4 bytes). However, only the value sum counters contain useful information about the elements of the sparsified model updates. The frequency counters and key sum counters are auxiliary information for the decompress process of IBLT.

We then decide the configuration m and W. First, we decide the counter number m. To ensure that the actual compression ratio of IBLT is r, one needs to choose $3s \cdot m = \frac{d}{r}$, solving the equation yields $m = \frac{d}{9r}$. Second, we decide the local sampling number W. When the modified IBLT table used in the KVSAgg-Minmax aggregates the sparsified gradients from K clients, the number of the union of the indices from K different clients is less than $K \cdot W$. According to the theory of KVSAgg, to ensure the aggregated IBLT can be perfectly decoded, one need to make sure that $1.25KW \leq m$ [29]. Solving the inequality yields $W \leq \frac{m}{11.25Kr}$. Similarly, we can figure out the configuration of W and m for KVSAgg-GSpar, which is exactly the same as that of KVSAGG-Minmax.

APPENDIX B
PROOF OF THEOREMS IN SECTION IV-C

Our theoretical analysis follows this roadmap: First, we represent the compression and decompression operations of QSRHT Sketch using matrix operations, which aligns with recent works on linear sketches [40], [66]. Next, we establish the unbiasedness and linearity properties of QSRHT Sketch based on its matrix representation. The notations in use can be referred to in Table II.

Probabilistic quantization. We first define the probabilistic quantization function on a vector. We define the probabilistic quantization of a vector $\mathbf{g} = (g_1,\ g_2,\ ...\ g_d) \in \mathbb{R}^d$ as

$$\mathcal{Q}(\mathbf{g}) = (Q(g_1),\ Q(g_2),\ ...\ Q(g_d))\,, \tag{3}$$

where Q is the probabilistic quantization function on a single scalar (see Section IV-A).

Matrix representation. We then define $\mathbf{R}$ and $\mathbf{D}$ as the corresponding matrices of the random hash functions R and D used in QSRHT Sketch, where:

- $\mathbf{R} \in \mathbb{R}^{m \times d}$ is a matrix which satisfies $\forall 0 \leq i \leq m-1, 0 \leq j \leq d-1$

$$\mathbf{R}_{i,j} = \begin{cases} 1, & \text{if } R(i) = j \\ 0, & \text{else} \end{cases}$$

- $\mathbf{D} \in \mathbb{R}^{d \times d}$ is a diagonal matrix (i.e. only the main diagonal contains non-zero elements) such that for all $0 \leq i \leq d-1$, $\mathbf{D}_{i,i} = D(i)$.

Note that the range of hash function $D(\cdot)$ is $\{-1, 1\}$, and therefore the matrix $\mathbf{D}$ satisfies $\mathbf{D} \cdot \mathbf{D} = \mathbf{I}_d$. With the above notations, the compression operation of QSRHT Sketch can be represented as follows. For a given model update vector $\mathbf{g} \in \mathbb{R}^d$:

$$Comp(\mathbf{g}) = \mathbf{R}\mathcal{Q}(\alpha\mathbf{HD} \cdot \mathbf{g}). \tag{4}$$

Similarly, the decompression operation of QSRHT Sketch can be represented as follows. For a given compressed array $\mathbf{h} \in \mathbb{R}^m$:

$$Decomp(\mathbf{h}) = \frac{d}{m\alpha}\mathbf{D}^T\mathbf{H}^T\mathbf{R}^T\mathbf{h} = \frac{d}{m\alpha}\mathbf{DHR}^T\mathbf{h}. \tag{5}$$

For the sake of simplicity, we denote the combination of $Comp(\cdot)$ and $Decomp(\cdot)$ operation as

$$\mathcal{C}_\alpha(\mathbf{g}) = Decomp(Comp(\mathbf{g})) = \frac{d}{m}\mathbf{DHR}^T\mathbf{R}\frac{\mathcal{Q}(\alpha \cdot \mathbf{HDg})}{\alpha}.$$

Lemmas. We present Lemma 1–Lemma 4, which help prove Theorem 1–Theorem 4. We simply explain the meaning of the lemmas. First, Lemma 1 demonstrates that $\mathbf{R} \in \mathbb{R}^{m \times d}$ exhibits similarity to an orthogonal matrix (i.e. the transpose of the matrix is the inverse of the matrix) in expectation. Lemma 1 is used in the proof of Theorem 1. Second, Lemma 2 and Lemma 3 show the unbiasedness and bounded variance property of probabilistic quantization, respectively. Third, Lemma 4 is a proposition of Lemma 3. Lemma 2 and Lemma 3 will be used to prove Theorem 1 and Theorem 3.

Lemma 1. *For the matrix* $\mathbf{R}$*, we have* $\mathbb{E}[\mathbf{R}^T\mathbf{R}] = \frac{m}{d}\mathbf{I_d}$.

Lemma 2. *For any* $\alpha \in \mathbb{R}, \mathbf{g} \in \mathbb{R}^d$, $\mathbb{E}\left[\frac{\mathcal{Q}(\alpha \cdot \mathbf{g})}{\alpha}\right] = \mathbf{g}$.

Lemma 3. *For any* $\alpha \in \mathbb{R}, \mathbf{g} \in \mathbb{R}^d$,

$$\mathbb{E}\left[\left\|\frac{\mathcal{Q}(\alpha \cdot \mathbf{g})}{\alpha} - \mathbf{g}\right\|^2\right] = \frac{d}{4\alpha^2}.$$

Lemma 4. *For any* $\alpha \in \mathbb{R}, \mathbf{g} \in \mathbb{R}^d$,

$$\mathbb{E}\left[\left\|\frac{\mathcal{Q}(\alpha \cdot \mathbf{g})}{\alpha}\right\|^2\right] = \frac{d}{4\alpha^2} + \|g\|^2.$$

We only prove Lemma 1, and the proofs of Lemma 2 and Lemma 3 can be refered to [42].

Proof of Lemma 1. Denote $\mathbf{A} = \mathbf{R}^T\mathbf{R}$. Then for $0 \le i \le m-1, 0 \le j \in d-1$, the element of $\mathbf{A}$ can be represented as $\mathbf{A}_{i,j} = \sum_{k=0}^{m-1} \mathbf{R}_{k,i}\mathbf{R}_{k,j}$. For $i \ne j$, we can get that $\mathbf{A}_{i,j} = 0$ based on the definition of $\mathbf{R}$. For $i = j$,

$$\mathbb{E}\left[\mathbf{A}_{i,i}\right] = \sum_{k=0}^{m-1} \mathbb{E}\left[\mathbb{I}[R(k) = i]\right] = \sum_{k=0}^{m-1} Pr[R(k) = i]$$

where $\mathbb{I}\left[R(k) = i\right]$ and $Pr\left[R(k) = i\right]$ represent the indicator function and the probability of the event $\{R(k) = i\}$, respectively. They satisfy $Pr\left[R(k) = i\right] = \mathbb{E}\left[\mathbb{I}[R(k)] = i\right]$. Since the hash function R maps each element $0 \le k \le m-1$ to $\{0, 1, ...d-1\}$ uniformly at random, For any $0 \le k \le m-1$ and $0 \le i \le d-1$, $Pr[R(k) = i] = \frac{1}{d}$. Therefore, $\mathbb{E}\left[\mathbf{A}_{i,i}\right] = \sum_{k=0}^{m-1} \frac{1}{d} = \frac{m}{d}$. In summary, we can conclude that

$$\mathbb{E}\left[\mathbf{A}_{i,j}\right] = \begin{cases} \frac{m}{d}, & i = j, \\ 0, & i \ne j, \end{cases}$$

which completes the proof. □

A. Proof of Theorem 1

Proof. The unbiasedness of the QSRHT sketch can be derived from the unbiasedness of both probabilistic quantization and SRHT. Specifically, we have:

$$\begin{aligned}
\mathbb{E}[\mathcal{C}_\alpha(\mathbf{g})] &= \mathbb{E}\left[\frac{d}{m}\mathbf{DHR}^T\mathbf{R}\mathbb{E}[\frac{\mathcal{Q}(\alpha \cdot \mathbf{HDg})}{\alpha}|\mathbf{D}, \mathbf{g}]\right] \\
&= \mathbb{E}\left[\frac{d}{m}\mathbf{DHR}^T\mathbf{RHDg}\right] \\
&= \mathbb{E}\left[\frac{d}{m}\mathbf{DH}\mathbb{E}[\mathbf{R}^T\mathbf{R}]\mathbf{HDg}\right] \\
&= \mathbb{E}\left[\frac{d}{m}\mathbf{DH}\frac{m}{d}\mathbf{I_d}\mathbf{HDg}\right] \\
&= \mathbf{g}.
\end{aligned}$$

The second equation holds because of Lemma 2. The third equation holds based on Lemma 1. And the last equation holds because $\mathbf{H} \cdot \mathbf{H} = \mathbf{I_d}$ and $\mathbf{D} \cdot \mathbf{D} = \mathbf{I_d}$. □

B. Proof of Theorem 2

Proof of Theorem 2.

$$\begin{aligned}
\mathbb{E}[\mathcal{C}_\alpha(\mathbf{g}_1 + \mathbf{g}_2)] &= \mathbf{g}_1 + \mathbf{g}_2 \\
&= \mathbb{E}[\mathcal{C}_\alpha(\mathbf{g}_1)] + \mathbb{E}[\mathcal{C}_\alpha(\mathbf{g}_2)] \\
&= \mathbb{E}[\mathcal{C}_\alpha(\mathbf{g}_1) + \mathcal{C}_\alpha(\mathbf{g}_2)],
\end{aligned}$$

where the first equation and the second equation hold because of Theorem 1. □

C. Proof of Theorem 3

Before we prove Theorem 3, we first prove the following lemma.

Lemma 5. *Suppose that* $\mathbf{R} \in \mathbb{R}^{m \times d}$ *is the given random matrix of QSRHT Sketch, then for any random vector* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ *which are indepedent of R. The following bound holds:*

$$\mathbb{E}\left[\langle \mathbf{R}^T\mathbf{Rx}, \mathbf{R}^T\mathbf{Ry}\rangle\right] = c_0\mathbb{E}[\langle \mathbf{x},\ \mathbf{y}\rangle], \tag{6}$$

where $c_0 := \frac{m}{d} + \frac{m(m-1)}{d^2}$.

Proof. Denote $\mathbf{A} = \mathbf{R}^T\mathbf{R}$, then $\mathbf{A}_{ij} = \sum_{k=1}^{m} \mathbf{R}_{ki}\mathbf{R}_{kj}$.

Recall the definition of the matrix $\mathbf{R}$, it is easy to observe that:

$$\mathbf{A}_{i,j} = \begin{cases} 0, & i \neq j \\ \sum_{k=1}^{m} \mathbf{R}_{ki}, & i = j. \end{cases} \tag{7}$$

Then we can calculate the expectation of $\mathbf{A}_{ii}^2$ as:

$$\begin{aligned} \mathbb{E}\left[\mathbf{A}_{ii}^2\right] &= \mathbb{E}\left[\sum_{k=1}^{m} \mathbf{R}_{ki} + \sum_{k_1 \neq k_2} \mathbf{R}_{k_1 i}\mathbf{R}_{k_2 i}\right] \\ &= \mathbb{E}\left[\sum_{k=1}^{m} \mathbf{R}_{ki}\right] + \sum_{k_1 \neq k_2} \mathbb{E}\left[\mathbf{R}_{k_1 i}\mathbf{R}_{k_2 i}\right] \\ &= \frac{m}{d} + \frac{m(m-1)}{d^2} \\ &= c_0 \end{aligned}$$

Therefore, we can express the left-hand side of Equation (6) as

$$\begin{aligned} \mathbb{E}[\langle \mathbf{R}^T\mathbf{R}\mathbf{x},\ \mathbf{R}^T\mathbf{R}\mathbf{y}\rangle] &= \sum_{i=1}^{d} \mathbb{E}[\mathbf{A}_{ii}^2 x_i y_i] \\ &= \sum_{i=1}^{d} \mathbb{E}\left[\mathbb{E}\left[\mathbf{A}_{ii}^2 \middle| \mathbf{x}, \mathbf{y}\right] x_i y_i\right] \\ &= \sum_{i=1}^{d} \mathbb{E}[c_0 x_i y_i] \\ &= c_0 \mathbb{E}[\langle \mathbf{x},\ \mathbf{y}\rangle] \end{aligned}$$

This completes the proof. □

Proof of Theorem 3. For simplicity, we denote $\mathbf{y} = \frac{\mathcal{Q}(\alpha \cdot \mathbf{HDg})}{\alpha}$. The error can be decomposed as

$$\begin{aligned} \mathbb{E}\left[\left\|\mathcal{C}_\alpha(\mathbf{g}) - \mathbf{g}\right\|^2\right] = \mathbb{E}\left[\left\langle \frac{d}{m}\mathbf{DHR}^T\mathbf{Ry}, \frac{d}{m}\mathbf{DHR}^T\mathbf{Ry} \right\rangle\right] \\ - 2 \cdot \mathbb{E}\left[\left\langle \frac{d}{m}\mathbf{DHR}^T\mathbf{Ry}, \mathbf{g} \right\rangle\right] + \|\mathbf{g}\|^2 \end{aligned} \tag{8}$$

For any $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$, we have

$$\langle \mathbf{DHu}, \mathbf{DHv} \rangle = \langle u, v \rangle . \tag{9}$$

That is because:

$$\begin{aligned} &\langle \mathbf{DHu}, \mathbf{DHv} \rangle \\ &= \mathbf{u}^T\mathbf{H}^T\mathbf{D}^T\mathbf{DHv} \\ &= \mathbf{u}^T\mathbf{HDDHv} \\ &= \mathbf{u}^T\mathbf{v} \\ &= \langle \mathbf{u}, \mathbf{v} \rangle . \end{aligned}$$

We use the fact that $\mathbf{D}^T = \mathbf{D}$, $\mathbf{H}^T = \mathbf{H}$, $\mathbf{D} \cdot \mathbf{D} = \mathbf{I}_d$, and $\mathbf{H} \cdot \mathbf{H} = \mathbf{I}_d$.

Therefore, the first term of Equation (8) can be simplified as:

$$\begin{aligned}
&\mathbb{E}\left[\left\langle \frac{d}{m}\mathbf{D}\mathbf{H}\mathbf{R}^T\mathbf{R}\mathbf{y}, \frac{d}{m}\mathbf{D}\mathbf{H}\mathbf{R}^T\mathbf{R}\mathbf{y}\right\rangle\right] \\
&= \left(\frac{d}{m}\right)^2 \mathbb{E}\left[\left\langle \mathbf{R}^T\mathbf{R}\mathbf{y}, \mathbf{R}^T\mathbf{R}\mathbf{y}\right\rangle\right] \\
&= \left(\frac{d}{m}\right)^2 \mathbb{E}\left[\mathbb{E}\left[\left\langle \mathbf{R}^T\mathbf{R}\mathbf{y}, \mathbf{R}^T\mathbf{R}\mathbf{y}\right\rangle | \mathbf{y}\right]\right] \\
&= \left(\frac{d}{m}\right)^2 \left(\frac{m}{d} + \frac{m(m-1)}{d^2}\right) \mathbb{E}\left[\|\mathbf{y}\|^2\right] \\
&= \left(\frac{d}{m} + \frac{m-1}{m}\right) \mathbb{E}\left[\left\| \frac{\mathcal{Q}(\alpha \cdot \mathbf{H}\mathbf{D}\mathbf{g})}{\alpha} \right\|^2\right] \\
&= \left(\frac{d}{m} + \frac{m-1}{m}\right) (\frac{d}{4\alpha^2} + \mathbb{E}\left[\|\mathbf{H}\mathbf{D}\mathbf{g}\|^2\right]) \\
&= \left(\frac{d}{m} + \frac{m-1}{m}\right) (\frac{d}{4\alpha^2} + \|\mathbf{g}\|^2).
\end{aligned}$$

The first equation holds because Equation (9). The second equation is the property of conditional expectation. The third equation holds because of Lemma 5. The fourth equation holds because of Lemma 3.

The second term of Equation (8) can be simplified by:

$$\begin{aligned}
&\mathbb{E}\left[\left\langle \frac{d}{m}\mathbf{D}\mathbf{H}\mathbf{R}^T\mathbf{R}\mathbf{y}, \mathbf{g}\right\rangle\right] \\
&= \frac{d}{m}\mathbb{E}\left[\left\langle \mathbf{R}^T\mathbf{R}\mathbf{y}, \mathbf{H}\mathbf{D}\mathbf{g}\right\rangle\right] \\
&= \frac{d}{m}\mathbb{E}\left[\left\langle \mathbf{R}^T\mathbf{R}\mathbb{E}\left[\frac{\mathcal{Q}(\alpha \cdot \mathbf{H}\mathbf{D}\mathbf{g})}{\alpha} | \mathbf{H}\mathbf{D}\mathbf{g}, \mathbf{R}\right], \mathbf{H}\mathbf{D}\mathbf{g}\right\rangle\right] \\
&= \frac{d}{m}\mathbb{E}\left[\left\langle \mathbf{R}^T\mathbf{R}\mathbf{H}\mathbf{D}\mathbf{g}, \mathbf{H}\mathbf{D}\mathbf{g}\right\rangle\right] \\
&= \frac{d}{m}\mathbb{E}\left[\left\langle \mathbb{E}\left[\mathbf{R}^T\mathbf{R}\mathbf{H}\mathbf{D}\mathbf{g} | \mathbf{H}\mathbf{D}\mathbf{g}\right], \mathbf{H}\mathbf{D}\mathbf{g}\right\rangle\right] \\
&= \frac{d}{m}\mathbb{E}\left[\left\langle \mathbb{E}\left[\mathbf{R}^T\mathbf{R}\right]\mathbf{H}\mathbf{D}\mathbf{g}, \mathbf{H}\mathbf{D}\mathbf{g}\right\rangle\right] \\
&= \frac{d}{m}\mathbb{E}\left[\left\langle \frac{m}{d}\mathbf{I}_d\mathbf{H}\mathbf{D}\mathbf{g}, \mathbf{H}\mathbf{D}\mathbf{g}\right\rangle\right] \\
&= \|\mathbf{g}\|^2.
\end{aligned}$$

The first equation holds because $D^T = D$ and $H^T = H$. The third equation holds because of Lemma 2. The fifth equation holds because of the conditional expectation. The sixth equation holds because we use Lemma 5. The last equation holds because $\langle \mathbf{H}\mathbf{D}\mathbf{g}, \mathbf{H}\mathbf{D}\mathbf{g}\rangle = \mathbf{g}^T\mathbf{D}^T\mathbf{H}^T\mathbf{H}\mathbf{D}\mathbf{g} = \langle \mathbf{g}, \mathbf{g}\rangle = \|\mathbf{g}\|^2$.

Plugging the first term and the second term back to Equation (8), we can get

$$\begin{aligned}
\mathbb{E}\left[\|\mathcal{C}_\alpha(\mathbf{g}) - \mathbf{g}\|^2\right] &= \left(\frac{d}{m} + \frac{m-1}{m}\right)(\frac{d}{4\alpha^2} + \|\mathbf{g}\|^2) - 2\|\mathbf{g}\|^2 + \|\mathbf{g}\|^2 \\
&= \left(\frac{d-1}{m}\right)\|\mathbf{g}\|^2 + \left(1 + \frac{d-1}{m}\right)\frac{d}{4\alpha^2} \\
&\leq \frac{d}{m}\|\mathbf{g}\|^2 + \left(1 + \frac{d}{m}\right)\frac{d}{4\alpha^2},
\end{aligned}$$

which completes the proof. □

Algorithm 1 The Full-participation FEDSSA.

1: Initialize $\mathbf{w}_0$.
2: **for** $i = 0, 1 \cdots T-1$ **do**
3: The server choose all clients and broadcast w_t.
4: **for** each client $i = 1, 2 \cdots K$ in parallel **do**
5: $\mathbf{w}_{t,0}^i = \mathbf{w}_t$.
6: **for** $k = 0, 1, \cdots E-1$ **do**
7: Compute $\mathbf{g}_{t,k}^i = \nabla F(\mathbf{w}_{t,k}^i, \mathcal{B}_{t,k}^i)$.
8: Local client update: $\mathbf{w}_{t,k+1}^i = \mathbf{w}_{t,k}^i - \eta \mathbf{g}_{t,k}^i$.
9: Let $\Delta_t^i = \frac{\mathbf{w}_{t,E}^i - \mathbf{w}_{t,0}^i}{\eta}$.
10: QSRHT Sketch compresse: $\tilde{\Delta}_t^i = Comp_t(\Delta_t^i)$.
11: Send $\tilde{\Delta}_t^i$ to server.
12: At server:
13: Receive $\tilde{\Delta}_t^i$, $1 \leq i \leq K$.
14: Let $\tilde{\Delta}_t = \frac{1}{K} \sum_{i=1}^K \tilde{\Delta}_t^i$.
15: Server decompress: $\tilde{g}_t = Decomp_t(\tilde{\Delta}_t)$.
16: Server update: $\mathbf{w}_{t+1} = \mathbf{w}_t + \eta \tilde{g}_t$.

APPENDIX C
PROOFS OF CONVERGENCE OF FEDSSA

A. The workflow of full participation FEDSSA.

We first introduce the workflow of full participation FEDSSA. Note that we omit the encryption, decryption, and periodic rehashing operations in FEDSSA. As shown in Algorithm 1, the FL procedure iterates for T rounds. In each round t, the server broadcasts the global model $\mathbf{w}_t$ to all clients (line 3). Then each client i saves the global model $\mathbf{w}_t$ and do E iteration for local model training. For client i, it computes the stochastic gradient $\mathbf{g}_{t,k}^i = \nabla F(\mathbf{w}_{t,k}^i, \mathcal{B}_{t,k}^i)$ in local iteration k and use it to refine it local model $\mathbf{w}_{t,k+1}^i = \mathbf{w}_{t,k}^i - \eta \mathbf{g}_{t,k}^i$ (lines 6-9). After finishing the local training, client i extracts local model update $\Delta_t^i = \frac{\mathbf{w}_{t,k+1}^i - \mathbf{w}_{t,0}^i}{\eta}$ (line 10). Then the client compresses it with the QSRHT Sketch with hash functions in round t to obtain compressed updates $\tilde{\Delta}_t^i = Comp_t(\Delta_t^i)$ and sends it to the server (line 11-12). The server receives compressed QSRHT Sketches from clients and aggregates to obtain aggregated QSRHT Sketch (lines 15-16). The server then decompresses the aggregated QSRHT Sketch to obtain model updates $\tilde{g}_t$ and uses it to refine the global model $\mathbf{w}_{t+1}$.

B. Proof of Theorem 4

Proof. This proof is mainly adapted from the Theorem1 in [47]. In each round t, since QSRHT Sketch uses different hash functions across rounds, we define the combination of compression and decompression by $\mathcal{C}_\alpha^t(.)$. The the global model updates obtained by server at round$t$ (i.e. $\tilde{g}_t$) can be represented by: $Decomp_t(\frac{1}{K}\sum_{i=1}^K Comp_t(\Delta_t^i)) = \frac{1}{K}\sum_{i=1}^K \mathcal{C}_\alpha^t(\Delta_t^i)$.

For convenience, we define $\bar{\Delta}_t \triangleq \frac{1}{K}\sum_{i=1}^K \mathcal{C}_\alpha^t(\Delta_t^i)$ and $\Delta_t = \frac{1}{K}\sum_{i=1}^K \Delta_t^i$. $\bar{\Delta}_t$ represents the sum of compressed model updates, while Δ_t represents the sum of original model updates.

It is clear that $\mathbb{E}\left[\bar{\Delta}_t\right] = \Delta_t = \frac{1}{K}\sum_{i=1}^K \Delta_t^i$.

Due to the smoothness in Assumption 1, taking expectation of $F\left(\mathrm{w}_{t+1}\right)$ over the randomness of the local stochastic gradient

of each client, the hash functions D_t, and R_t at communication round t, we have:

$$\begin{aligned}
&\mathbb{E}_t\left[F\left(\mathbf{w}_{t+1}\right)\right] \\
&\leq F\left(\mathbf{w}_t\right)+\left\langle \nabla F\left(\mathbf{w}_t\right), \mathbb{E}_t\left[\mathbf{w}_{t+1}-\mathbf{w}_t\right]\right\rangle \\
&+\frac{L}{2}\mathbb{E}_t\left[\left\|\mathbf{w}_{t+1}-\mathbf{w}_t\right\|^2\right] \\
&=F\left(\mathbf{w}_t\right)+\eta\left\langle \nabla F\left(\mathbf{w}_t\right), \mathbb{E}_t\left[\bar{\Delta}_t+E\nabla F\left(\mathbf{w}_t\right)-\eta E\nabla F\left(\mathbf{w}_t\right)\right]\right\rangle \\
&+\frac{L}{2}\eta^2\mathbb{E}_t\left[\left\|\bar{\Delta}_t\right\|^2\right] \\
&=F\left(\mathbf{w}_t\right)-\eta E\left\|\nabla F\left(\mathbf{w}_t\right)\right\|^2 \\
&+\eta\underbrace{\left\langle \nabla F\left(\mathbf{w}_t\right), \mathbb{E}_t\left[\bar{\Delta}_t+E\nabla F\left(\mathbf{w}_t\right)\right]\right\rangle}_{A_1} \\
&+\frac{L}{2}\eta^2\underbrace{\mathbb{E}_t\left[\left\|\bar{\Delta}_t\right\|^2\right]}_{A_2}.
\end{aligned} \tag{10}$$

Note that the term A_1 in inequality (10) can be bounded as follows:

$$\begin{aligned}
A_1&=\left\langle \nabla F\left(\mathbf{w}_t\right), \mathbb{E}_t\left[\bar{\Delta}_t+E\nabla F\left(\mathbf{w}_t\right)\right]\right\rangle \\
&=\left\langle \nabla F\left(\mathbf{w}_t\right), \mathbb{E}_t\left[\Delta_t+E\nabla F\left(\mathbf{w}_t\right)\right]\right\rangle \\
&=\left\langle \nabla F\left(\mathbf{w}_t\right), \mathbb{E}_t\left[-\frac{1}{K}\sum_{i=1}^{K}\sum_{k=0}^{E-1}\mathbf{g}_{t,k}^i+E\nabla F\left(x_t\right)\right]\right\rangle \\
&=\left\langle \nabla F\left(\mathbf{w}_t\right), \mathbb{E}_t\left[-\frac{1}{K}\sum_{i=1}^{K}\sum_{k=0}^{E-1}\nabla F_i\left(\mathbf{w}_{t,k}^i\right)+E\frac{1}{K}\sum_{i=1}^{K}\nabla F_i\left(\mathbf{w}_t\right)\right]\right\rangle \\
&=\left\langle \sqrt{E}\nabla F\left(\mathbf{w}_t\right), -\frac{1}{K\sqrt{E}}\mathbb{E}_t\sum_{i=1}^{K}\sum_{k=0}^{E-1}\left(\nabla F_i\left(\mathbf{w}_{t,k}^i\right)-\nabla F_i\left(\mathbf{w}_t\right)\right)\right\rangle \\
&\overset{(a1)}{=}\frac{E}{2}\left\|\nabla F\left(\mathbf{w}_t\right)\right\|^2+\frac{1}{2EK^2}\mathbb{E}_t\left\|\sum_{i=1}^{K}\sum_{k=0}^{E-1}\nabla F_i\left(\mathbf{w}_{t,k}^i\right)-\nabla F_i\left(\mathbf{w}_t\right)\right\|^2 \\
&-\frac{1}{2EK^2}\mathbb{E}_t\left\|\sum_{i=1}^{K}\sum_{k=0}^{E-1}\nabla F_i\left(\mathbf{w}_{t,k}^i\right)\right\|^2 \\
&\overset{(a2)}{\leq}\frac{E}{2}\left\|\nabla F\left(\mathbf{w}_t\right)\right\|^2+\frac{1}{2K}\sum_{i=1}^{K}\sum_{k=0}^{E-1}\mathbb{E}_t\left\|\nabla F_i\left(\mathbf{w}_{t,k}^i\right)-\nabla F_i\left(\mathbf{w}_t\right)\right\|^2 \\
&-\frac{1}{2EK^2}\mathbb{E}_t\left\|\sum_{i=1}^{K}\sum_{k=0}^{E-1}\nabla F_i\left(\mathbf{w}_{t,k}^i\right)\right\|^2 \\
&\overset{(a3)}{\leq}\frac{E}{2}\left\|\nabla F\left(\mathbf{w}_t\right)\right\|^2+\frac{L^2}{2K}\sum_{i=1}^{K}\sum_{k=0}^{E-1}\mathbb{E}_t\left\|\mathbf{w}_{t,k}^i-\mathbf{w}_t\right\|^2 \\
&-\frac{1}{2EK^2}\mathbb{E}_t\left\|\sum_{i=1}^{K}\sum_{k=0}^{E-1}\nabla F_i\left(\mathbf{w}_{t,k}^i\right)\right\|^2 \\
&\overset{(a4)}{\leq}E\left(\frac{1}{2}+15E^2\eta^2L^2\right)\left\|\nabla F\left(x_t\right)\right\|^2+\frac{5E^2\eta^2L^2}{2}\left(\sigma_L^2+6E\sigma_G^2\right) \\
&-\frac{1}{2EK^2}\mathbb{E}_t\left\|\sum_{i=1}^{K}\sum_{k=0}^{E-1}\nabla F_i\left(\mathbf{w}_{t,k}^i\right)\right\|^2,
\end{aligned} \tag{11}$$

where $(a1)$ follows from that

$$\langle \mathbf{w}, \mathbf{y}\rangle=\frac{1}{2}\left[\|\mathbf{w}\|^2+\|\mathbf{y}\|^2-\|\mathbf{w}-\mathbf{y}\|^2\right]$$

for $\mathbf{w} = \sqrt{E}\nabla F\left(\mathbf{w}_t\right)$ and

$$\mathbf{y} = -\frac{1}{K\sqrt{E}}\sum_{i=1}^{K}\sum_{k=0}^{E-1}\left(\nabla F_i\left(\mathbf{w}_{t,k}^i\right) - \nabla F_i\left(\mathbf{w}_t\right)\right),$$

$(a2)$ is due to that

$$\mathbb{E}\left[\left\|x_1 + \cdots + x_n\right\|^2\right] \leq n\mathbb{E}\left[\left\|x_1\right\|^2 + \cdots + \left\|x_n\right\|^2\right],$$

$(a3)$ is due to Assumption 1 and (a4) follows from Lemma 2

The term A_2 in (1) can be bounded as:

$$\begin{aligned}
A_2 &= \mathbb{E}_t\left[\left\|\bar{\Delta}_t\right\|^2\right] \\
&= \mathbb{E}_t\left[\left\|\frac{1}{K}\sum_{i=1}^{K}\mathcal{C}_\alpha^t(\Delta_t^i)\right\|^2\right] \\
&= \frac{1}{K^2}\mathbb{E}_t\left[\sum_{i=1}^{K}\left\|\mathcal{C}_\alpha^t(\Delta_t^i)\right\|^2\right] \\
&+ \frac{1}{K^2}\mathbb{E}_t\left[\sum_{i\neq j}\langle\mathcal{C}_\alpha^t(\Delta_t^i), \mathcal{C}_\alpha^t(\Delta_t^j)\rangle\right] \\
&= \frac{1}{K^2}\mathbb{E}_t\left[\sum_{i=1}^{K}\left(1+\frac{d-1}{m}\right)\left(\frac{d}{4\alpha^2} + \left\|\Delta_t^i\right\|^2\right)\right] \\
&+ \frac{1}{K^2}\left(1+\frac{d-1}{m}\right)\mathbb{E}_t\left[\sum_{i\neq j}\langle\Delta_t^i, \Delta_t^j\rangle\right] \\
&= \frac{(d+m-1)d}{4mK\alpha^2} + \left(1+\frac{d-1}{m}\right)\frac{1}{K^2}\mathbb{E}_t\left[\left\|\frac{1}{K}\sum_{i=1}^{K}\Delta_t^i\right\|^2\right] \\
&\leq \frac{(d+m)d}{4mK\alpha^2} + \left(1+\frac{d}{m}\right)\frac{1}{K^2}\mathbb{E}_t\left[\left\|\sum_{i=1}^{K}\sum_{k=0}^{E-1}\mathbf{g}_{t,k}^i\right\|^2\right] \\
&\overset{(a5)}{=} \frac{(d+m)d}{4mK\alpha^2} + \left(1+\frac{d}{m}\right)\frac{1}{K^2}\mathbb{E}_t\left[\left\|\sum_{i=1}^{K}\sum_{k=0}^{E-1}\left(\mathbf{g}_{t,k}^i - \nabla F_i\left(\mathbf{w}_{t,k}^i\right)\right)\right\|^2\right] \\
&+ \left(1+\frac{d}{m}\right)\frac{1}{K^2}\mathbb{E}_t\left\|\sum_{i=1}^{K}\sum_{k=0}^{E-1}\nabla F_i\left(\mathbf{w}_{t,k}^i\right)\right\|^2 \\
&\overset{(a6)}{\leq} \frac{(d+m)d}{4mK\alpha^2} + \left(1+\frac{d}{m}\right)\frac{E}{K}\sigma_L^2 \\
&+ \left(1+\frac{d}{m}\right)\frac{1}{K^2}\mathbb{E}_t\left\|\sum_{i=1}^{K}\sum_{k=0}^{E-1}\nabla F_i\left(\mathbf{w}_{t,k}^i\right)\right\|^2,
\end{aligned} \tag{12}$$

$(a5)$ follows from the fact that

$$\mathbb{E}\left[\|\mathbf{w}\|^2\right] = \mathbb{E}\left[\|\mathbf{w} - \mathbb{E}[\mathbf{w}]\|^2\right] + \|\mathbb{E}[\mathbf{w}]\|^2\right],$$

and $(a6)$ is due to the bounded variance assumption in Assumption 3 and the fact that

$$\mathbb{E}\left[\left\|x_1 + \cdots + x_n\right\|^2\right] = \mathbb{E}\left[\left\|x_1\right\|^2 + \cdots + \left\|x_n\right\|^2\right]$$

if x_i' s are independent with zero mean and $\mathbb{E}\left[\mathbf{g}_{t,j}^i\right] = \nabla F_i\left(\mathbf{w}_{t,j}^i\right)$.

Substituting the inequalities in (11) of A_1 and (12) of A_2 into inequality (10), we have:

$$
\begin{aligned}
&\mathbb{E}_t\left[F\left(\mathbf{w}_{t+1}\right)\right] \\
&\leq F\left(\mathbf{w}_t\right)-\eta E\left\|\nabla F\left(\mathbf{w}_t\right)\right\|^2 \\
&+\eta \underbrace{<\nabla F\left(\mathbf{w}_t\right), \mathbb{E}_t\left[\bar{\Delta}_t+\eta E \nabla F\left(\mathbf{w}_t\right)\right]>}_{A_1}+\frac{L}{2} \eta^2 \underbrace{\mathbb{E}_t\left[\left\|\bar{\Delta}_t\right\|^2\right]}_{A_2} \\
&\leq F\left(\mathbf{w}_t\right)-\eta E\left(\frac{1}{2}-15 E^2 \eta^2 L^2\right)\left\|\nabla F\left(\mathbf{w}_t\right)\right\|^2 \\
&+\frac{L E \eta^2}{2 K}\left(1+\frac{d-1}{m}\right) \sigma_L^2+\frac{L \eta^2(d+m) d}{4 m K \alpha^2} \\
&+\frac{5 E^2 \eta^3 L^2}{2}\left(\sigma_L^2+6 E \sigma_G^2\right) \\
&-\left(\frac{\eta}{2 E K^2}-\left(1+\frac{d-1}{m}\right) \frac{L \eta^2}{2 K^2}\right) \mathbb{E}_t\left\|\sum_{i=1}^K \sum_{k=0}^{E-1} \nabla F_i\left(\mathbf{w}_{t, k}^i\right)\right\|^2 \\
&\overset{(a7)}{\leq} F\left(\mathbf{w}_t\right)-\eta E\left(\frac{1}{2}-5 E^2 \eta^2 L^2\right)\left\|\nabla F\left(\mathbf{w}_t\right)\right\|^2 \\
&+\frac{L E \eta^2}{2 K}\left(1+\frac{d-1}{m}\right) \sigma_L^2+\frac{L \eta^2(d+m) d}{4 m K \alpha^2}+\frac{5 E^2 \eta^3 L^2}{2}\left(\sigma_L^2+6 E \sigma_G^2\right) \\
&\overset{(a8)}{\leq} F\left(\mathbf{w}_t\right)-c \eta E\left\|\nabla F\left(\mathbf{w}_t\right)\right\|^2 \\
&+\frac{L E \eta^2}{2 K}\left(1+\frac{d-1}{m}\right) \sigma_L^2+\frac{L \eta^2(d+m) d}{4 m K \alpha^2}+\frac{5 E^2 \eta^3 L^2}{2}\left(\sigma_L^2+6 E \sigma_G^2\right)
\end{aligned}
$$

$(a7)$ follows from

$$
\left(\frac{\eta}{2 E K^2}-\left(1+\frac{d-1}{m}\right) \frac{L \eta^2}{2 K^2}\right) \geq 0
$$

if $\eta \leq \frac{1}{E L\left(1+\frac{d-1}{m}\right)}$.

$(a8)$ holds because there exists a constant $c>0$ satisfying $\left(\frac{1}{2}-15 E^2 \eta^2 L^2\right)>c>0$ if $\eta<\frac{1}{\sqrt{30} E L}$.

Rearranging and summing from $t=0, \cdots, T-1$, we have:

$$
\begin{aligned}
\sum_{t=0}^{T-1} c \eta E \mathbb{E}\left[\left\|\nabla F\left(\mathbf{w}_t\right)\right\|^2\right] \leq & F\left(\mathbf{w}_0\right)-F\left(\mathbf{w}_T\right) \\
&+T \eta E\left[\frac{L \eta}{2 K}\left(1+\frac{d-1}{m}\right) \sigma_L^2\right] \\
&+T \eta E\left[\frac{5 E \eta^2 L^2}{2}\left(\sigma_L^2+6 E \sigma_G^2\right)\right] \\
&+T \eta \frac{L \eta(d+m) d}{4 m K \alpha^2}
\end{aligned}
$$

It implies that $\min_{t \in[T]} \mathbb{E}\left\|\nabla F\left(\mathbf{w}_t\right)\right\|_2^2 \leq \frac{F_0-F_*}{c \eta E T}+\Phi$, where

$$
\Phi=\frac{1}{c}\left[\frac{L \eta(1+r) d}{4 E K \alpha^2}+\frac{L \eta}{2 K}\left(1+\frac{d-1}{m}\right) \sigma_L^2+\frac{5 E \eta^2 L^2}{2}\left(\sigma_L^2+6 E \sigma_G^2\right)\right] .
$$

This completes the proof. □