PICASO: Secure Aggregation for Federated Learning with Minimal Synchronization

Anonymous Author(s)
Affiliation
Address
email

Abstract

Preventing private data leakage is crucial in federated learning. Existing secure aggregation (SA) protocols, which are the core protocols for privacy-preserving federated learning, require clients to synchronize at multiple points, meaning they must wait for other clients to send their messages before proceeding. This synchronization ensures that inputs can be aggregated without compromising privacy, while also accounting for client dropouts and message delays.

This work presents PICASO, abbreviated from Per Iteration Client At most Synchronizes Once, a novel SA protocol minimizing synchronization overhead in privacy preserving federated learning, aligning its communication pattern more closely with that of non-private federated learning. PICASO outperforms previous works like SecAgg, SecAgg+, MicroSecAgg, and Flamingo with server runtime under 1 second for large clients. PICASO demonstrates viability by training various models on different datasets.

We also detail extensions to PICASO to achieve various improvements over state-of-the-art algorithms in two key areas - detecting and removing malicious clients, and secure aggregation for heterogeneous datasets. Overall, PICASO presents an efficient, secure, and flexible federated learning solution minimizing synchronization needs.

1 Introduction

Federated learning (FL) enables collaborative machine learning without sharing client data, by aggregating local model updates at a central coordinator. However, recent works show that training data can still be compromised from model updates alone [51, 82, 71], making secure aggregation (SA) crucial for privacy-preserving FL. SA computes the sum of user inputs while keeping individual inputs private. Like in traditional FL, the server typically selects a set of n clients for each iteration with the server repeating the process until the model converges. Typically, a new set of clients are chosen per iteration Note that the number of clients n can range from a hundred to tens of millions [58] and similarly the number of model parameters m can scale to millions [13]. The goal is to securely train a global model. Critically, for SA protocols to help with federated learning, the underlying protocol is needed to be robust to client dropouts. Furthermore, SA algorithms typically work over integers or field elements while the weights produced by an ML model are floating point values. Therefore, one often needs to quantize the weights and show that the model produced by the SA protocols still preserve accuracy. Existing SA algorithms broadly rely on Differential Privacy (DP) [44], Homomorphic Encryption (HE) [75, 96], or secure multiparty computation techniques [8, 13, 7, 68, 66, 86, 83, 98, 56, 94, 67, 63].

SecAgg [13] introduced a practical solution for privacy-preserving horizontal federated learning. The protocol's core idea involves pairwise masking seeds $s_{u,v}$ shared between clients $u, v \in \mathcal{U}$, where $\mathcal{U}$ is the set of all users. Each client u masks its input using $\sum_{v<u} s_{u,v} - \sum_{v>u} s_{u,v}$. Note that a client

$v < u$ would generate its mask by subtracting $s_{u,v}$. Therefore, it is easy to see that the pairwise masks cancel out in aggregation, i.e., $\sum_{u \in \mathcal{U}} (\sum_{v<u} s_{u,v} - \sum_{v>u} s_{u,v}) = 0$. While a particular user u maybe offline, the remaining clients would still have used their pairwise mask with u in the aggregation. Therefore, $s_{u,v}$ for an offline u and any online v needs to be secret shared with other clients to allow the server to reconstruct $s_{u,v}$ and then remove the mask. Unfortunately, a server could label an online client u as offline which would give the server $s_{u,v}$, allowing it to unmask an online user's inputs. Therefore, a client also uses a self-mask s_u to mask its inputs. This s_u is also secret-shared and is reconstructed should u be online. As is obvious, SecAgg involves multiple rounds of computation such as to establish pairwise masks, secret share the masks, sending the masked inputs, and then reconstructing the sum. Therefore, for n clients and a vector of size m, the protocol requires $O(n^2m)$ computation on the part of the aggregator, $O(mn)$ for each client. Subsequent works have focused on reducing the complexity through various assumptions and techniques. Here the vector m can be viewed as the number of parameters in the model, i.e., the inputs to the secure aggregation algorithm.

This work aims to address a critical scalability issue with existing SA algorithms. Prior works often, including SecAgg that was described above, require clients to synchronize their participation with others, an artifact of techniques where clients mask their inputs but must share masks with a quorum of clients to facilitate unmasking, if the client was unavailable later. This expensive ritual of sharing masks induces a bottleneck absent in non-private training, where clients simply train the model and send updates without additional synchronization.

Our main contribution is PICASO, a secure aggregation protocol where each client synchronizes at most once per training iteration. The key idea is that in iteration ℓ, client i masks its input x_i by computing $y_i = x_i + \mathsf{GenerateMask}(k_i, \ell)$, where k_i is its *private* key. Client i sends y_i to the server and $\mathsf{GenerateMask}(k_i, \ell)$ to a separate "collector" party. Our model supports dynamic selection of collector. They are stateless and run a deterministic computation, simpler than the server's own computation. A simple way of choosing a collector would be to use a randomness beacon [37] and the Algorithm 1 from Flamingo [68, Lines 2-5]. The collector aggregates the masks from all clients and sends their sum to the server. The server then reconstructs $\sum_i x_i$ using the masked inputs and the sum of masks. Unlike SecAgg (and its subsequent works), PICASO does not require sending masks to multiple parties or secret sharing, reducing synchronization overhead. It only needs to synchronize to identify the collector for that iteration. Looking ahead, the collector acts can be viewed as a single "decryptor" [68] or "committee member" [63], receiving information from clients, condensing it, and communicating the result to the server for secure aggregation (Figure 1a). In other words, PICASO utilizes one intermediate party while SecAgg employed n with subsequent works employing $\log n$ intermediate parties.

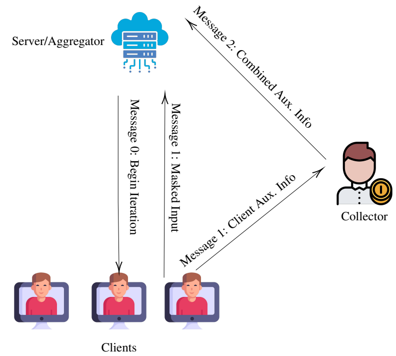
Asymptotically, PICASO's client computation cost is $O(m)$, where m is the input vector length (e.g., the number of weights of the model), while the server and collector computation cost is $O(mn)$, where n is the number of clients (Table 2). PICASO offers several attractive features:

- Dropout tolerance: Any number of selected clients can opt out without increasing computational burden on remaining clients or requiring additional interaction.
- Collusion resistance: Privacy of honest users' inputs is preserved even if an adversary corrupts any number of clients and the aggregator.
- Scalability and dynamism: New clients can join without an expensive setup phase, needing only public parameters and the aggregator's iteration key.
- Enhanced privacy: Input privacy is maintained against both collector and aggregator, provided they do not collude. The collector can change in each iteration, facilitated by a randomness beacon which in tern prevents server manipulation.

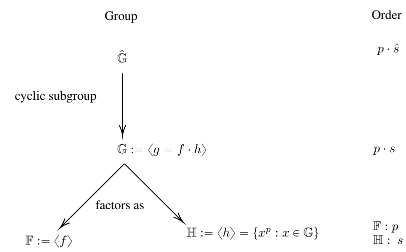
We also microbenchmark PICASO, comparing with the state-of-the-art secure aggregation algorithms to demonstrate competitive performance. For example, PICASO's server computation time is $< 1s$, even for large number of clients besting prior work. We also conduct extensive experiments on FL benchmark datasets to demonstrate that PICASO preserves performance, while guaranteeing privacy.

Further, PICASO can easily be extended to offer:

- a constant-round protocol to detect and remove malicious clients (i.e., sending inconsistent or incorrect messages), improving on the state-of-the-art ACORN which requires $O(\log n)$ rounds where n is the number of clients.



(a) The PICASO system model operates in iterations. Each iteration begins with the server sending a message to initiate the process (Message 0). In response, clients train the model on their local data, obtain updates, and mask the input. *Concurrently*, clients communicate with both the server and the collector (Message 1): masked input is sent to the server, while auxiliary information is transmitted to the collector. Upon receiving information from all clients, the collector combines these into a single value. Finally, this consolidated data is sent to the server (Message 2), concluding the iteration.



(b) A brief overview of the class group framework we employ. Here, $\widehat{\mathbb{G}}$ is group, whose order is $\widehat{s} \cdot p$, such that $\widehat{s}$ and p are co-prime. Further, s divides $\widehat{s}$ and is the order of the group $\mathbb{G}$, which is generated by g and is denoted as $\mathbb{G} := \langle g \rangle$. Similarly, $\mathbb{H}$ is a subgroup of $\mathbb{G}$, generated by h whose order is s, while $\mathbb{F}$ has order p and is generated by f. We have $g = f \cdot h$. Further, $\widehat{s}$ (and s) is unknown but an upper bound $\bar{s}$ is known. The last property we will rely on is that discrete logarithm is efficient in the subgroup $\mathbb{F}$.

Figure 1: The backbone of PICASO - the communication system model and the CL framework.

- a secure aggregation protocol supporting heterogeneous datasets via robust stochastic averaging [64], which improves upon DReS-FL [80] as DReS-FL requires the entire dataset to be secret shared among clients, which we avoid.

1.1 Related Work

Secure Aggregation Using Differential Privacy and Homomorphic Encryption. A simple approach to differential privacy (DP) is local DP [38], where clients add noise to their data before sending it to the server. This has been deployed by major tech companies [40, 1]. However, research shows that such data perturbation may reduce accuracy. Our techniques can be composed with DP solutions, using secure aggregation (SA) algorithms to mask noisy local inputs [57]. Meanwhile, BatchCrypt [96] employed homomorphic encryption (HE), building on earlier work. However, it required all clients to use the same key, posing a significant privacy risk.

SA using Multiparty Computation. Secure multiparty computation (MPC) preserves privacy and accuracy by computing over encrypted data. Early works on Private Stream Aggregation [81] focused on secure summing of streaming data. Following SecAggBonawitz et al. [13], Federated Learning protocols with dropout resilience were developed, but multiple interaction rounds increased dropout risk. Subsequent works [8, 7, 68, 66, 86, 83, 98, 56, 94, 67, 63] have focused on reducing the number of intermediate parties to $\log n$, or reusing the masked secret sharing across multiple iterations to reduce round count. This is summarized in Table 1 where we compare various protocols with respect to the following properties: (a) the number of rounds of interaction, (b) whether it can tolerate client dropouts, (c) on whether the aggregate value can be efficiently recovered, (d) public setup for security assumption, and (e) number of intermediate parties needed to help with the aggregation. Multi-server settings [33, 4] face challenges with long inputs due to increased communication and computation demands. Our approach assumes no server-collector collusion, a weaker assumption as the collector changes each iteration and performs less computation.

2 System Model and Relevant Background

We consider a federated learning framework, as shown in Figure 1a. There exist n clients, with each client C_i owning a dataset D_i. The server holds the ML model Θ. In FL, the server first senders Θ to clients, and each client trains its local dataset on D_i to generate the updated weights m_i. Meanwhile,

Table 1: Comparison of various Secure Aggregation Algorithms based on MPC.

	# Rounds	Dropout Resilience	Efficient Aggregation	Public Setup	# Additional Parties
[81]	1	✗	✗	✓	0
[55]	1	✗	✓	✗	0
[9]	1	✗	✗	✗	0
[62]	1	✓	✓	✗	1
SecAgg[13]	3	✓	✓	✓	n
SecAgg+[8]	3	✓	✓	✓	$\log n$
MicroSecAgg[49]	1+2	✓	✗*	✓	$\log n$
LERNA[63]	1+2	✓	✓	✓	$\log n$
Flamingo[68]	1+2	✓	✓	✓	$\log n$
PICASO	1	✓	✓	✓	1

Table 2: Comparison of asymptotic complexity of some secure aggregation protocols. Note that in PICASO, the collector performs $O(mn)$ computation and communication.

	Client		Server	
	Computation	Communication	Computation	Communication
SecAgg[13]	$O(mn + n^2)$	$O(m + n)$	$O(mn^2)$	$O(mn + n^2)$
SecAgg+[8]	$O(m \log n + \log^2 n)$	$O(m + \log n)$	$O(n \log^2 n + mn \log n)$	$O(mn + n \log n)$
SASH[66]	$O(m + n^2)$	$O(m + n)$	$O(m + n^2)$	$O(mn + n^2)$
PICASO	$O(m)$	$O(m)$	$O(mn)$	$O(mn)$

the server computes the average of these model updates $\{m_i\}$ to update its global model to Θ'. In the next iteration Θ' is sent back for the next update. The goal is to use the collector to ensure that the weights m_i remain secret while still allowing the server to compute the average, and thereby the new model Θ'.

Threat Model. Our threat model follows the long line of prior works whereby an adversary can: (a) corrupt the server *or* the collector, (b) corrupt clients which enables the adversary to choose the client inputs for an iteration. The goal is to ensure that the honest users' inputs remain private with only their sum being leaked. Our protocols are described in this setting, like all prior work. Note that prior works such as Flamingo [68], or LERNA [63] did not guarantee privacy when all the intermediate parties collude with each other. Similarly, we allow for the collector to corrupt clients and guarantee the security against a corrupted collector. If the server and collector collude at an iteration, however, there is no privacy for that iteration.[1] Importantly, our collector can change from every iteration to iteration and is selected by a randomness beacon using an algorithm similar to how the set is chosen in Flamingo [68, Algorithm 1]. This is similar to how validators are chosen in some proof of stake blockchains [45].

Modeling Security. We model security against both a corrupt server and a corrupt collector. A corrupt server can adaptively compromise clients and collude with them, issue arbitrary encryption messages for honest parties in any iteration, and receive the collector's combined information at each iteration, but cannot corrupt the collector. The adversary selects honest clients $H_1, \ldots, H_t$ and provides two input sets: $\{x_1, \ldots, x_t\}$ and $\{x'_1, \ldots, x'_t\}$, where $\sum x_i = \sum x'_i$. The challenger randomly selects and encrypts one set for the target time period τ. The adversary's goal is to determine which set was chosen with probability significantly exceeding 1/2.

Meanwhile, for privacy against a corrupt collector, the adversary receives individual communication sent by the clients to the collector. It can corrupt clients and also issue encryption queries, as before. It cannot corrupt the server but can adaptively issue the above queries. The challenge is the same as for a corrupt server - to distinguish between honest user inputs. Since it does not receive the final result, the challenge sets need not have the same sum.

CL **Framework.** Cryptographic protocols often use cyclic groups G of prime order q, generated by g_q, i.e., $G := \{1, g_q, \ldots, g_q^{q-1}\}$. The Discrete Logarithm (DL) Assumption [2] states that given $X \in G$, finding x where $g_q^x = X$ is computationally infeasible.[2] The Decisional Diffie-Hellman

[1] In such a case the server can learn the individual client model updates. To protect against such an attack the best that the clients can do is to add differential private noise to their updates.

[2] Small x are recoverable, as in [49, 81, 9].

(DDH) Assumption [14] posits that given g_q, g_q^x, g_q^y, distinguishing g_q^{xy} from a random element in G is computationally infeasible.

The CL Framework [24, 27–29, 17] introduces the idea of a composite order group, where the order is unknown, but there is a subgroup of known prime order where the discrete logarithm computation is easy. This framework utilizes the group where DL is easy to ensure correctness and eventual message recovery, and the group where DL is hard to achieve security. The framework is summarized in Figure 1b.

The security property relies on a modified DDH assumption (Definition 1) involving indistinguishability between elements from groups $\mathbb{G}$ and $\mathbb{H}$ within a composite order group. While their orders are unknown, upper bounds exist. The input space is in $\mathbb{F}$, and the key space in $\mathbb{H}$. Distributions $\mathcal{D}_G$ and $\mathcal{D}_H$ are based on these upper bounds, with $\mathcal{D}_G$ (resp. $\mathcal{D}_H$) being statistically indistinguishable from $\mathbb{G}$'s (resp. $\mathbb{H}$'s) exponent space. Typically, $\mathcal{D}_H := 0, \ldots, B-1$ where $B = 2^{40} \cdot \bar{s}$ where $\bar{s}$ is the upperbound of the order of $\mathbb{H}$, shown by [89] to be 2^{-40}-close to uniform. The security property relies on a modification of the DDH assumption (see Definition 1), where the indistinguishability is between elements from two different groups, $\mathbb{G}$ and $\mathbb{H}$, within the composite order group. However, the orders of $\mathbb{G}$ and $\mathbb{H}$ are unknown, but there are upper bounds on their orders.

Definition 1 (DDH-f Assumption)**.** *Let $(\widehat{\mathbb{G}}, \mathbb{G}, \mathbb{H}, \mathbb{F}, \bar{s})$ be the class group as defined in Figure 1b. Then, the following two distributions are computationally indistinguishable, i.e., there is no "efficient" attacker who can distinguish whether it is the first or the second distribution that a sampled value comes from, with a probability greater than half. Here, $x, y \leftarrow_{\$} \mathcal{D}_H, u \leftarrow_{\$} \mathbb{Z}/p\mathbb{Z}$*

$$\{(h, h^x, h^y, h^{xy}\} \approx_c \{(h, h^x, h^y, h^{xy} \cdot f^u\}$$

We refer the readers to Bouvier *et al.* [16] and Tucker [89] for a detailed exposition on class groups, techniques, and its extensive use in cryptography. They also survey the utility of CL framework in building other cryptographic primitives.

3 PICASO

In this section, we present an efficient privacy-preserving aggregation scheme called PICASO where each iteration involves a client speaking at most once. We begin by describing an additively homomorphic masking algorithm. We then instantiate this in the CL framework. This is a generalization of the version presented in the introduction. We then present our complete description of PICASO and formally prove it secure in the random oracle model.

3.1 Homomorphic Masking Algorithm

Let k_i denote the secret key of Client i. Let k_0 denote the secret key of the aggregator. Further, for iteration ℓ, let $\mathsf{pk}_{i,\ell}$ (resp. $\mathsf{pk}_{0,\ell}$) denote the public key of client i (resp. the server) for iteration ℓ. Then, we require the following properties of our algorithm $\mathsf{GenMask}$:

- The masking function can be computed using two different ways, i.e., $\mathsf{GenMask}(\mathsf{pk}_{i,\ell}, \mathsf{k}_0) = \mathsf{GenMask}(\mathsf{pk}_{0,\ell}, \mathsf{k}_i)$
- Homomorphic over public key space, i.e., $\prod_{i=1}^{n} \mathsf{GenMask}(\mathsf{pk}_{i,\ell}, \mathsf{k}_0) = \mathsf{GenMask}(\prod_{i=1}^{n} \mathsf{pk}_{j,\ell}, \mathsf{k}_0)$

Further, we require that the generated mask is pseudorandom, i.e., $\mathsf{GenMask}(\mathsf{pk}_{0,\ell}, \mathsf{k}_i)$ appears random provided the adversary cannot compute the mask on its own which requires the knowledge of $(\mathsf{pk}_{0,\ell}, \mathsf{k}_i)$ or $(\mathsf{pk}_{i,\ell}, \mathsf{k}_0)$.

Construction 1 (Homomorphic Masking Algorithm)**.** Let $\mathcal{H} : \{0,1\}^* \to \mathbb{H}$ be a hash function that maps strings to the unknown order subgroup of $\mathbb{G}$.[3] Then, for secret key $\mathsf{k}_i \leftarrow_{\$} \mathcal{D}_H$ for $i = 0, \ldots, n$, we can define, for iteration ℓ, $\mathsf{pk}_{i,\ell} := \mathcal{H}(\ell)^{\mathsf{k}_i}$ and the mask value as $\mathsf{mask}_{i,\ell} := \mathcal{H}(\ell)^{\mathsf{k}_0 \cdot \mathsf{k}_i}$

We now show that the construction satisfies the required properties:

[3] Note that for our purposes we can simply begin by hashing the input to an element in $\mathcal{D}_H$, and then raising the group generator h to this value. This is because the knowledge of the discrete logarithm is not detrimental. However, [77] present additional methods to hash into a group of unknown order, in a way that the discrete logarithm is unknown.

- $\mathsf{mask}_{i,\ell} = \mathcal{H}(\ell)^{\mathsf{k}_0 \cdot \mathsf{k}_i} = \mathsf{pk}_{i,\ell}^{\mathsf{k}_0} = \mathsf{pk}_{0,\ell}^{\mathsf{k}_i}$
- $\mathsf{mask}_{i,\ell} \cdot \mathsf{mask}_{j,\ell} = \mathcal{H}(\ell)^{\mathsf{k}_0(\mathsf{k}_i + \mathsf{k}_j)} = (\mathsf{pk}_{i,\ell} \cdot \mathsf{pk}_{j,\ell})^{\mathsf{k}_0}$

For a particular iteration ℓ, an adversary is either given $\mathcal{H}(\ell), \mathsf{pk}_{0,\ell}, \mathsf{pk}_{i,\ell}, \mathsf{mask}_{i,\ell}$ or $\mathcal{H}(\ell), \mathsf{pk}_{0,\ell}, \mathsf{pk}_{i,\ell}, U$ where $U \leftarrow\$ \mathbb{G}$. This follows from the DDH-f assumption, which we define in Definition 1. Looking ahead, this pseudorandom mask will be used to mask the client input and thereby guaranteeing privacy.

3.2 Formal Description of PICASO

We first informally describe the protocol. At iteration ℓ, the server sends a message identifying clients who are participating in that round of interaction. In this message, it also includes its iteration public key $\mathsf{pk}_{0,\ell}$. Client i, with its input $x_{i,\ell}$, first encodes it as $f^{x_{i,\ell}}$. Recall that f is the generator of the cyclic, prime-order group $\mathbb{F}$ where discrete logarithm is easy, i.e., given this encoding, there exists an efficient algorithm that outputs $x_{i,\ell}$. Once encoded, it computes the mask $\mathsf{mask}_{i,\ell} = \mathsf{GenMask}(\mathsf{pk}_{0,\ell}, \mathsf{k}_i)$. It sends to the server the masked input $\mathsf{ct}_{i,\ell} = \mathsf{mask}_{i,\ell} \cdot f^{x_{i,\ell}}$. Meanwhile, it also sends to the collector $\mathsf{pk}_{i,\ell}$.

The collector simply multiplies all of the clients' iteration public keys to compute $\mathsf{AUX}_\ell = \prod \mathsf{pk}_{i,\ell}$. AUX_ℓ is sent to the server. The server does the following: multiplies all of the masked inputs $\prod \mathsf{ct}_{i,\ell}$ and divides it by $\mathsf{GenMask}(\mathsf{AUX}_\ell, \mathsf{k}_0)$. It then applies the efficient discrete logarithm to compute the aggregate. Formally, we present in Construction 2.

Construction 2 (PICASO Protocol for iteration ℓ)**.** The protocol description is as follows:

- **One-Time Setup Phase:**
  Transparent Setup is executed and outputs $\mathsf{pp} = (\widehat{\mathbb{G}}, \mathbb{F}, p, \mathcal{D}_H, \mathcal{D}_G, \bar{s}, g, h, f, \mathcal{H} : \{0,1\}^* \to \mathbb{H})$
- **Begin Iteration:** Server, with key k_0, computes $\mathsf{pk}_{0,\ell} = \mathcal{H}(\ell)^{\mathsf{k}_0}$ and sends to the chosen clients and collector.
- **Encryption Phase:** Each online client $C_i \in \mathcal{OL}_\ell$ with key k_i and input $x_{i,\ell}$ does the following:
  - Compute $\mathsf{mask}_{i,\ell} := \mathsf{GenMask}(\mathsf{pk}_{0,\ell}, \mathsf{k}_i)$
  - Compute masked input $\mathsf{ct}_{i,\ell} = f^{x_{i,\ell}} \cdot \mathsf{mask}_{i,\ell}$
  - Compute public key $\mathsf{pk}_{i,\ell} = \mathcal{H}(\ell)^{\mathsf{k}_i}$
  - $\mathbf{C_i}$ →**Server**: $\mathsf{ct}_{i,\ell}$
  - $\mathbf{C_i}$ →**Collector**: $\mathsf{pk}_{i,\ell}$
- **Collection Phase:** Collector computes $\mathsf{AUX}_\ell = \prod_{i \in \mathcal{OL}_\ell} \mathsf{pk}_{i,\ell}$.
  Collector→Server: AUX_ℓ
- **Aggregation Phase**: Server computes:
  - Compute $Y_\ell := \prod_{i \in \mathcal{OL}_\ell} \mathsf{ct}_{i,\ell}$
  - Compute $X_\ell := \mathsf{GenMask}(\mathsf{AUX}_\ell, \mathsf{k}_0)$
  - Compute $Sum_\ell := Y_\ell / X_\ell$
  - Take discrete log of Sum_ℓ, which is efficient.

We omit the proof due to space constraints. However, the intuition for security comes from the fact that: (a) the honest user's key is chosen by the honest user and is unknown to the adversary, (b) for such a random key, the mask generated is indeed pseudorandom under the DDH-f assumption, and (c) such a pseudorandom mask will blind the honest client's inputs.

Remark 1. Observe that it is possible that the client's communication to either the server or the collector is dropped due to network issues. In this situation, the collector's information relayed to the server does not yield correct aggregate. To handle such situation, the server and the collector can engage in one additional round of communication, per iteration. In this round, the collector first sends a list of clients from whom it has received communication. The server respond with the intersection of the collector's list with its own list of clients. Finally, the collector "collects" only with respect to this set of clients.

Remark 2. Note that each $\mathsf{pk}_{i,\ell}$ is pseudorandom, if k_i is unknown. However, masking only with $\mathsf{pk}_{i,\ell}$ is insufficient for security against the collector. This is because the collector receives the masked input (masked by $\mathsf{pk}_{i,\ell}$) and $\mathsf{pk}_{i,\ell}$ for ever honest client i. Therefore, the collector can easily recover the input. This prompts the need for a server's iteration public key, generated as a function of its secret key k_0.

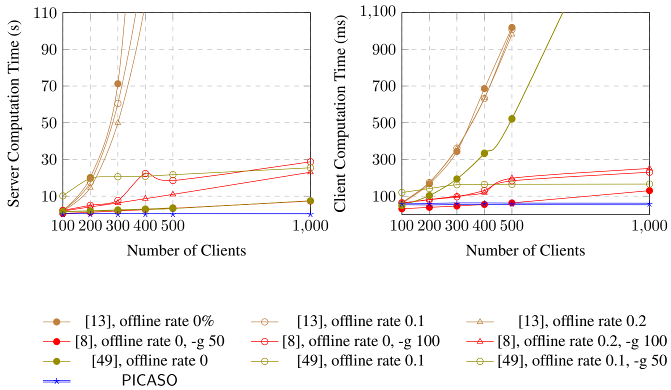


Figure 2: Measure of Server and Client Computation Time as a function of number of clients across various aggregation algorithms.

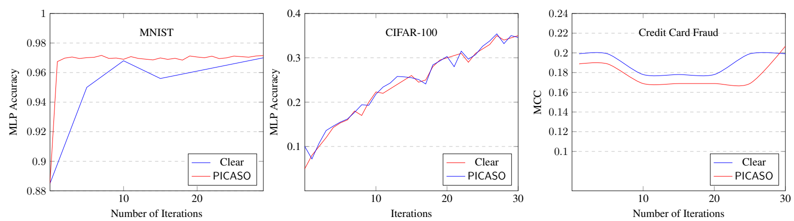


Figure 3: Performance Measurement of PICASO for FL Tasks.

4 Experiments

In this section, we perform different experimentation to demonstrate the efficiency, accuracy, and privacy of PICASO. All our experiments were carried out on an Apple M1 Pro CPU with 16 GB of unified memory, without any multi-threading or related parallelization. More details can be found in Section B.

Microbenchmarking Secure Aggregation. We benchmark the client and server computation time of our protocol against existing state-of-the-art solutions, including [13], [8], MicroSecAgg [49], and PICASO. Additionally, we compare our results with specific parameter choices from prior work, such as grouping operations (clients share inputs with 50 or 100 parties) and offline rate (parties can go offline during the protocol). These settings are not applicable to PICASO. Our reported timing is taken as a mean of 20 iterations.

As shown in Figure 2, our client computation time is significantly better than [13] and [8], and comparable to MicroSecAgg. However, unlike MicroSecAgg, our protocol does not incur offline waiting times due to multiple rounds of participation. For instance, when there are 100 clients, MicroSecAgg requires at least 30ms of offline time, which increases with more clients. Additionally, MicroSecAgg limits input size to achieve server efficiency, supporting only small model updates or quantized large model updates. Figure 2 demonstrates that PICASO's server running time is under 1 second, thanks to a single-round protocol with efficient aggregate recovery. This outperforms all other protocols. Any additional communication required to capture Remark 1 has a negligible impact on computation time, as it only involves gathering the list of clients and communicating with the collector. SASH [66] combines the secure aggregation protocol SecAgg [13] with a seed-homomorphic PRG to enhance efficiency for encrypting large input vectors. However, their performance is dominated by SecAgg, which we significantly outperform. Combining SASH with PICASO could achieve efficient round communication and improved server computation time, optimizing for input size scaling. Finally, PICASO requires 56 bytes of bandwidth for each of the following: server public key, masked input, while requiring 32 bytes for client's iteration public key sent to the collector, and information sent to the server from the collector.

Benchmarking FL Models. To demonstrate PICASO's viability for federated learning, we use it to train various models. We train a logistic regression model on Kaggle Credit Card Fraud

Dataset [76]. Figure 3 shows PICASO's MCC versus clear learning for varying clients and iterations. With the accuracy multiplier, PICASO's MCC is very close to clear learning and even outperforms sometimes. The highly unbalanced dataset demonstrates PICASO can achieve strong performance even in challenging real-world scenarios. We then train a vanilla multi-layer perceptron (MLP) classifier on three datasets: MNIST, CIFAR-100. The details of the datasets, including quantization and license can be found in Table 4. The MLP accuracy, as a function of the iteration, is plotted in Figure 3. Our experiments demonstrate that PICASO preserves accuracy, while ensuring the privacy of client data. Note that vanilla MLP classifiers do not typically offer good performance for CIFAR datasets, but note that the goal of our experiments was to show that PICASO does not impact accuracy.

5 Extensions to PICASO

We defer additional extensions to Section D, due to space constraints.

5.1 Robust PICASO

PICASO requires clients to send iteration public keys to the collector and masked inputs to the server, potentially allowing malicious actors to disrupt aggregation by using inconsistent keys. While secure aggregation has been widely studied, less focus has been on detecting and mitigating malicious behavior. Prior works in this domain are limited to:

- ACORN [7]: Offers a constant-round version detecting malicious behavior (aborting on detection) and a non-constant round version removing malicious inputs.
- RoFL [67] and ACORN: Use zero-knowledge proofs (e.g., Bulletproofs [19] and improvements [43]) to prevent malicious input injection.

The latter requires that the secure aggregation algorithm still proceeds, after having removed malicious clients. Indeed, PICASO ensures privacy of inputs, even if the server is corrupt and chooses to corrupt various users. Similarly, against corrupt collector who can corrupt users and inject messages into the system. In this section, we show how to augment PICASO to detect and remove malicious clients as described above. In Algorithm 3, we only present the additional proving steps by the client and the verification steps for the collector. In the construction, $\mathscr{C}$ is representative of the challenge space and integer A is chosen as a function of the size of $\mathscr{C}$. We refer the reader to [17, §5.2] for details about the proof system and its correctness. Here, A is set to be an integer such that the size of challenge space $C := |\mathscr{C}|$ is negligibly small when compared to A ,i.e., C/A is negligible.

Signatures can be employed to ensure the collector transmits only client-authenticated information to the server, mitigating malicious collector behavior. Our protocol can be enhanced with range-proof techniques from ACORN [7]. Notably, our inputs are encoded in prime-order subgroup $\mathbb{F}$, which can be composed with standard Pedersen commitments [74] using a prime order cyclic subgroup G where the DDH assumption holds.

Construction 3 (Additional Steps in Robust-PICASO)**.** We assume that there is an hash function $\mathbb{H} : \{0,1\}^* \to \mathscr{C}$. Here, $A := 2^{40} \cdot |\mathcal{D}_H| \cdot C$ and $[A] := \{0, \ldots, A-1\}$. We set C to be 2^{128}.

Proof Generation: Each online client C_i

- Sample $r_k \leftarrow\$ [A], r_x \leftarrow\$ \{0, \ldots, p-1\}$
- Compute $t_1 := \mathcal{H}(\ell)^{r_k}, t_2 := \mathsf{pk}_{0,\ell}^{r_k} \cdot f^{r_x}$
- Compute $ch := \mathbb{H}(\ell, \mathsf{pk}_{i,\ell}, \mathsf{ct}_{i,\ell}, t_1, t_2, \mathsf{pk}_{0,\ell})$
- Compute $s_k := r_k + ch \cdot \mathsf{k}_i, s_x := r_x + ch \cdot x_{i,\ell} \bmod p$
- Set $\mathsf{proof}_i := (s_k, s_x, ch)$
- $\mathbf{C_i} \to$**Server**: $\mathsf{ct}_{i,\ell}$
- $\mathbf{C_i} \to$**Collector**: $\mathsf{pk}_{i,\ell}, \mathsf{ct}_{i,\ell}, \mathsf{proof}_i$

Proof Verification: Collector does: For client i in $\mathcal{OL}_\ell$:

- Receive: $(\mathsf{pk}_{i,\ell}, \mathsf{ct}_{i,\ell}, \mathsf{proof}_i = (s_k, s_x, ch))$
- Compute $t_2' := \mathsf{pk}_{0,\ell}^{s_k} \cdot f^{s_x} \cdot \mathsf{ct}_{i,\ell}^{-ch}$
- Compute $t_1' := (\mathcal{H}(\ell))^{s_k} \cdot \mathcal{H}(\ell)^{-ch}$
- Compute $ch' := \mathbb{H}(\ell, \mathsf{pk}_{i,\ell}, \mathsf{ct}_{i,\ell}, t_1', t_2', \mathsf{pk}_{0,\ell})$
- **if** $ch \neq ch'$ **then**
  $\mathcal{OL}_\ell := \mathcal{OL}_\ell \backslash \{i\}$
  Add $(\mathsf{proof}_i, \mathsf{ct}_{i,\ell}, \mathsf{pk}_{i,\ell})$ to list $\mathcal{M}$
- Compute $\mathsf{AUX}_\ell := \prod_{i \in \mathcal{OL}_\ell} \mathsf{pk}_{i,\ell}$
- **Collector→Server**:
  $\mathsf{AUX}_\ell, \{\mathsf{ct}_{i,\ell}\}_{i \in \mathcal{OL}_\ell}, \mathcal{M}$

5.2 Byzantine-Robust Stochastic Aggregation and Heterogeneous Datasets

Heterogeneity in Data Distribution. Data-centric methods [97, 72, 54] aim to align local and global distributions while preserving privacy, using techniques like sharing raw, synthesized, or augmented data. However, these approaches may compromise local data privacy [79]. Privacy-preserving machine learning can be achieved through secret sharing schemes such as homomorphic encryption (HE) [42, 46] and multiparty computation (MPC) [70]. However, HE is computationally expensive, and MPC faces scalability issues. Recent frameworks [85] utilize Lagrange coding and polynomial approximations to address these challenges in federated learning settings. RSA [64] is a class of stochastic sub-gradient methods for distributed learning robust to Byzantine workers. It mitigates the effects of incorrect messages due to malicious behavior, communication failures, or uneven data distribution by incorporating a regularization term in the objective function. At each iteration k, clients compute parameter updates based on local data, prior local models, and global parameters. The client and server updates are:

$$\begin{aligned}\text{Client: } x_i^{k+1} &= x_i^k - \eta^k \left(\nabla F(x_i^k, \xi_i^k) + \lambda \text{sign}(x_i^k - w^k)\right)\\ \text{Server: } w^{k+1} &= w^k - \eta^k \left(\nabla f_0(w^k) + \lambda \sum_{i\in[n]} \text{sign}(w^k - x_i^k)\right)\end{aligned}$$

where η is the learning rate, ξ is a local dataset sample, $F(\cdot,\cdot)$ is the loss function, $f_{\ell_2}(\cdot)$ is the robust regularization term, λ weights the robustness term, sign is element-wise, and $[n]$ is the client set.

Secure Aggregation with RSA. As pointed out by Franzese *et al.* [41], the only information needed by the server to aggregate is $\text{sign}(w^k - x_i^k)$. In other words, the clients simply need to supply the server with a vector with elements in $\{-1, 1\}$. Furthermore, representing -1 as a 0 yields the following property: $2 \cdot \sum_{i=1}^n v_i - n = \sum_{i=1}^n u_i$ where $u_i \in \{-1, 1\}$ and $v_i = 0$ iff $u_i = -1$. In summary, the server has to perform aggregation over binary vectors. PICASO can be used to perform this securely, with only the client having to prove that the masked input is either 0 or 1. Such a proof is efficient and we describe below. We present the additional steps to be performed by the clients and the server in Construction 4, where the client proving that it has encrypted either a value of 0 or a value of 1. This is an adaptation of Groth and Kohlweiss [48] to the CL Framework. We omit the proof due to space constraints but it follows earlier results from Braun *et al.* [17].

Construction 4 (Secure, Byzantine-Robust Secure Aggregation with PICASO)**.** We assume that there is an hash function $\mathbb{H} : \{0, 1\}^* \to \mathscr{C}$. Here, $A := 2^{40} \cdot |\mathcal{D}_H| \cdot C$ and $[A] := \{0, \ldots, A - 1\}$.

Proof Generation: Each online client C_i is encrypting $x_{i,\ell} \in \{0, 1\}$ where $\mathsf{ct}_{i,\ell} := \mathsf{pk}_{0,\ell}^{\mathsf{k}_i} \cdot f^{x_{i,\ell}}$

- Sample $r_k, r'_k \leftarrow_\$ [A], r_x \leftarrow_\$ \{0, \ldots, p - 1\}$
- Compute $t_1 := \mathsf{pk}_{0,\ell}^{r_k} \cdot f^{r_x}, t_2 := \mathsf{pk}_{0,\ell}^{r'_k} \cdot f^{r_x \cdot x_{i,\ell}}$
- Compute $ch := \mathbb{H}(\ell, \mathsf{pk}_{i,\ell}, \mathsf{ct}_{i,\ell}, t_1, t_2, \mathsf{pk}_{0,\ell})$
- Compute $s_x := r_x + ch \cdot x_{i,\ell} \mod p$
- Compute $s_k := r_k + ch \cdot \mathsf{k}_i, , s'_k := r'_k + (ch - s_x) \cdot \mathsf{k}_i$
- Set $\mathsf{proof}_i := (s_k, s'_k, s_x, ch)$
- $\mathbf{C_i}$ →**Server**: $\mathsf{ct}_{i,\ell}, \mathsf{proof}_i$
- $\mathbf{C_i}$ →**Collector**: $\mathsf{pk}_{i,\ell}$

Proof Verification: Server does: For client i in $\mathcal{OL}_\ell$:

- Receive: $(\mathsf{pk}_{i,\ell}, \mathsf{ct}_{i,\ell}, \mathsf{proof}_i = (s_k, s'_k, s_x, ch))$
- Compute $t'_1 := \mathsf{ct}_{i,\ell}^{-ch} \cdot f^{s_x} \cdot \mathsf{pk}_{0,\ell}^{s_k}$
- Compute $t'_2 := \mathsf{ct}_{i,\ell}^{s_x - ch} \cdot \mathsf{pk}_{0,\ell}^{s'_k}$
- Compute $ch' := \mathbb{H}(\ell, \mathsf{pk}_{i,\ell}, \mathsf{ct}_{i,\ell}, t'_1, t'_2, \mathsf{pk}_{0,\ell})$
- **if** $ch \neq ch'$ **then** $\mathcal{OL}_\ell := \mathcal{OL}_\ell \backslash \{i\}$

6 Conclusion

We present PICASO which is a secure aggregation protocol where client only has to synchronize once. This is an improvement over existing secure aggregation protocols. We also demonstrate that our protocol has a competitive performance over these protocols, and outperforms several of them. Finally, we show that PICASO preserves accuracy, while guaranteeing privacy. Our encryption time increasing proportionally with the length of vector. While this is expected, our use of group expoentiations makes the process slower. A possible direction for future research is to apply the SASH framework [66] with PICASO, which reduces number of group exponentiations. Despite limitations we believe PICASO significantly solves the scalability problem, while ensuring privacy. Its possible extensions improve on several state-of-the-art protocols.

References

[1] Apple machine learning research. URL https://machinelearning.apple.com/research/learning-with-privacy-at-scale.

[2] Discrete logarithm. URL en.wikipedia.org/wiki/Discrete_logarithm. Wikipedia.

[3] D. Abram, I. Damgård, C. Orlandi, and P. Scholl. An algebraic framework for silent preprocessing with trustless setup and active security. In Y. Dodis and T. Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part IV*, volume 13510 of *Lecture Notes in Computer Science*, pages 421–452. Springer, Heidelberg, Aug. 2022. doi: 10.1007/978-3-031-15985-5_15.

[4] S. Addanki, K. Garbe, E. Jaffe, R. Ostrovsky, and A. Polychroniadou. Prio+: Privacy preserving aggregate statistics vianbsp;boolean shares. In *Security and Cryptography for Networks: 13th International Conference, SCN 2022, Amalfi (SA), Italy, September 12–14, 2022, Proceedings*, page 516–539, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-14790-6. doi: 10.1007/978-3-031-14791-3_23. URL https://doi.org/10.1007/978-3-031-14791-3_23.

[5] A. Arun, C. Ganesh, S. Lokam, T. Mopuri, and S. Sridhar. Dew: Transparent constant-sized zkSNARKs. Cryptology ePrint Archive, Report 2022/419, 2022. https://eprint.iacr.org/2022/419.

[6] T. Attema, I. Cascudo, R. Cramer, I. B. Damgård, and D. Escudero. Vector commitments over rings and compressed σ-protocols. Cryptology ePrint Archive, Report 2022/181, 2022. https://eprint.iacr.org/2022/181.

[7] J. Bell, A. Gascón, T. Lepoint, B. Li, S. Meiklejohn, M. Raykova, and C. Yun. ACORN: Input validation for secure aggregation. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 4805–4822, Anaheim, CA, Aug. 2023. USENIX Association. ISBN 978-1-939133-37-3. URL https://www.usenix.org/conference/usenixsecurity23/presentation/bell.

[8] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1253–1269, 2020.

[9] F. Benhamouda, M. Joye, and B. Libert. A new framework for privacy-preserving aggregation of time-series data. *ACM Trans. Inf. Syst. Secur.*, 18(3), mar 2016. ISSN 1094-9224. doi: 10.1145/2873069. URL https://doi.org/10.1145/2873069.

[10] W. Beullens, T. Kleinjung, and F. Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In S. D. Galbraith and S. Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 227–247. Springer, Heidelberg, Dec. 2019. doi: 10.1007/978-3-030-34578-5_9.

[11] J.-F. Biasse. Improvements in the computation of ideal class groups of imaginary quadratic number fields, 2010. ISSN 1930-5346. URL /article/id/0151cba7-9512-448b-83ae-a272f0b836ce.

[12] J.-F. Biasse, M. J. Jacobson, and A. K. Silvester. Security estimates for quadratic field based cryptosystems. In R. Steinfeld and P. Hawkes, editors, *ACISP 10: 15th Australasian Conference on Information Security and Privacy*, volume 6168 of *Lecture Notes in Computer Science*, pages 233–247. Springer, Heidelberg, July 2010.

[13] 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 B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1175–1191. ACM, 2017. doi: 10.1145/3133956.3133982. URL https://doi.org/10.1145/3133956.3133982.

[14] D. Boneh. The decision diffie-hellman problem. In J. P. Buhler, editor, *Algorithmic Number Theory*, pages 48–63, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. ISBN 978-3-540-69113-6.

[15] D. Boneh, B. Bünz, and B. Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In A. Boldyreva and D. Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 561–586. Springer, Heidelberg, Aug. 2019. doi: 10.1007/978-3-030-26948-7_20.

[16] C. Bouvier, G. Castagnos, L. Imbert, and F. Laguillaumie. I want to ride my BICYCL : BICYCL implements cryptography in class groups. *J. Cryptol.*, 36(3):17, 2023. doi: 10.1007/s00145-023-09459-1. URL https://doi.org/10.1007/s00145-023-09459-1.

[17] L. Braun, I. Damgård, and C. Orlandi. Secure multiparty computation from threshold encryption based on class groups. In H. Handschuh and A. Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part I*, volume 14081 of *Lecture Notes in Computer Science*, pages 613–645. Springer, Heidelberg, Aug. 2023. doi: 10.1007/978-3-031-38557-5_20.

[18] J. Buchmann and H. C. Williams. A key-exchange system based on imaginary quadratic fields. *Journal of Cryptology*, 1(2):107–118, June 1988. doi: 10.1007/BF02351719.

[19] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018. doi: 10.1109/SP.2018.00020.

[20] B. Bünz, B. Fisch, and A. Szepieniec. Transparent SNARKs from DARK compilers. In A. Canteaut and Y. Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 677–706. Springer, Heidelberg, May 2020. doi: 10.1007/978-3-030-45721-1_24.

[21] D. Byrd, M. Hybinette, and T. H. Balch. Abides: Towards high-fidelity multi-agent market simulation. In *Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM-PADS '20, page 11–22, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450375924. doi: 10.1145/3384441.3395986. URL `https://doi.org/10.1145/3384441.3395986`.

[22] I. Cascudo and B. David. ALBATROSS: Publicly AttestabLe BATched Randomness based On Secret Sharing. In S. Moriai and H. Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part III*, volume 12493 of *Lecture Notes in Computer Science*, pages 311–341. Springer, Heidelberg, Dec. 2020. doi: 10.1007/978-3-030-64840-4_11.

[23] G. Castagnos and F. Laguillaumie. On the security of cryptosystems with quadratic decryption: The nicest cryptanalysis. In A. Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 260–277. Springer, Heidelberg, Apr. 2009. doi: 10.1007/978-3-642-01001-9_15.

[24] G. Castagnos and F. Laguillaumie. Linearly homomorphic encryption from DDH. In K. Nyberg, editor, *Topics in Cryptology – CT-RSA 2015*, volume 9048 of *Lecture Notes in Computer Science*, pages 487–505. Springer, Heidelberg, Apr. 2015. doi: 10.1007/978-3-319-16715-2_26.

[25] G. Castagnos, A. Joux, F. Laguillaumie, and P. Q. Nguyen. Factoring pq^2 with quadratic forms: Nice cryptanalyses. In M. Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 469–486. Springer, Heidelberg, Dec. 2009. doi: 10.1007/978-3-642-10366-7_28.

[26] G. Castagnos, L. Imbert, and F. Laguillaumie. Encryption switching protocols revisited: Switching modulo p. In J. Katz and H. Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 255–287. Springer, Heidelberg, Aug. 2017. doi: 10.1007/978-3-319-63688-7_9.

[27] G. Castagnos, F. Laguillaumie, and I. Tucker. Practical fully secure unrestricted inner product functional encryption modulo p. In T. Peyrin and S. Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 733–764. Springer, Heidelberg, Dec. 2018. doi: 10.1007/978-3-030-03329-3_25.

[28] G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker. Two-party ECDSA from hash proof systems and efficient instantiations. In A. Boldyreva and D. Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 191–221. Springer, Heidelberg, Aug. 2019. doi: 10.1007/978-3-030-26954-8_7.

[29] G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker. Bandwidth-efficient threshold EC-DSA. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, editors, *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 12111 of *Lecture Notes in Computer Science*, pages 266–296. Springer, Heidelberg, May 2020. doi: 10.1007/978-3-030-45388-6_10.

[30] Z. Chai, Y. Chen, L. Zhao, Y. Cheng, and H. Rangwala. Fedat: A communication-efficient federated learning method with asynchronous tiers under non-iid data. *arXiv preprint arXiv:2010.05958*, 2020.

[31] P. Chaidos and G. Couteau. Efficient designated-verifier non-interactive zero-knowledge proofs of knowledge. In J. B. Nielsen and V. Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 193–221. Springer, Heidelberg, Apr. / May 2018. doi: 10.1007/978-3-319-78372-7_7.

[32] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala. Asynchronous online federated learning for edge devices with non-iid data. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 15–24. IEEE, 2020.

[33] H. Corrigan-Gibbs and D. Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In A. Akella and J. Howell, editors, *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, pages 259–282. USENIX Association, 2017. URL `https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/corrigan-gibbs`.

[34] G. Couteau, M. Klooß, H. Lin, and M. Reichle. Efficient range proofs with transparent setup from bounded integer commitments. In A. Canteaut and F.-X. Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part III*, volume 12698 of *Lecture Notes in Computer Science*, pages 247–277. Springer, Heidelberg, Oct. 2021. doi: 10.1007/978-3-030-77883-5_9.

[35] G. Couteau, D. Goudarzi, M. Klooß, and M. Reichle. Sharp: Short relaxed range proofs. In H. Yin, A. Stavrou, C. Cremers, and E. Shi, editors, *ACM CCS 2022: 29th Conference on Computer and Communications Security*, pages 609–622. ACM Press, Nov. 2022. doi: 10.1145/3548606.3560628.

[36] Y. Deng, S. Ma, X. Zhang, H. Wang, X. Song, and X. Xie. Promise Σ-protocol: How to construct efficient threshold ECDSA from encryptions based on class groups. In M. Tibouchi and H. Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 557–586. Springer, Heidelberg, Dec. 2021. doi: 10.1007/978-3-030-92068-5_19.

[37] Drand. Drand/drand: a distributed randomness beacon daemon - go implementation. URL `https://github.com/drand/drand`.

[38] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, aug 2014. ISSN 1551-305X. doi: 10.1561/0400000042. URL `https://doi.org/10.1561/0400000042`.

[39] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985. doi: 10.1109/TIT.1985.1057074.

[40] Ú. Erlingsson, V. Pihur, and A. Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In G.-J. Ahn, M. Yung, and N. Li, editors, *ACM CCS 2014: 21st Conference on Computer and Communications Security*, pages 1054–1067. ACM Press, Nov. 2014. doi: 10.1145/2660267.2660348.

[41] N. Franzese, A. Dziedzic, C. A. Choquette-Choo, M. R. Thomas, M. A. Kaleem, S. Rabanser, C. Fang, S. Jha, N. Papernot, and X. Wang. Robust and actively secure serverless collaborative learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=SouroWC5Un`.

[42] C. Gentry. Fully homomorphic encryption using ideal lattices. *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[43] C. Gentry, S. Halevi, and V. Lyubashevsky. Practical non-interactive publicly verifiable secret sharing with thousands of parties. In O. Dunkelman and S. Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part I*, volume 13275 of *Lecture Notes in Computer Science*, pages 458–487. Springer, Heidelberg, May / June 2022. doi: 10.1007/978-3-031-06944-4_16.

[44] R. C. Geyer, T. Klein, and M. Nabi. Differentially private federated learning: A client level perspective. *CoRR*, abs/1712.07557, 2017. URL `http://arxiv.org/abs/1712.07557`.

[45] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, page 51–68, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450350853. doi: 10.1145/3132747.3132757. URL `https://doi.org/10.1145/3132747.3132757`.

[46] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International conference on machine learning*, pages 201–210. PMLR, 2016.

[47] N. Glaeser, M. Maffei, G. Malavolta, P. Moreno-Sanchez, E. Tairi, and S. A. Thyagarajan. Foundations of coin mixing services. Cryptology ePrint Archive, Report 2022/942, 2022. `https://eprint.iacr.org/2022/942`.

[48] J. Groth and M. Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 253–280. Springer, Heidelberg, Apr. 2015. doi: 10.1007/978-3-662-46803-6_9.

[49] Y. Guo, A. Polychroniadou, E. Shi, D. Byrd, and T. Balch. Microsecagg: Streamlined single-server secure aggregation. *Proceedings on Privacy Enhancing Technologies*, 2024(4):77–101, 2024. doi: 10.56553/popets-2024-0077. URL `https://doi.org/10.56553/popets-2024-0077`.

[50] J. L. Hafner and K. S. McCurley. A rigorous subexponential algorithm for computation of class groups. *Journal of the American mathematical society*, 2(4):837–850, 1989.

[51] B. Hitaj, G. Ateniese, and F. Pérez-Cruz. Deep models under the GAN: Information leakage from collaborative deep learning. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 603–618. ACM Press, Oct. / Nov. 2017. doi: 10.1145/3133956.3134012.

[52] D. Hühnlein, M. J. Jacobson Jr., S. Paulus, and T. Takagi. A cryptosystem based on non-maximal imaginary quadratic orders with fast decryption. In K. Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 294–307. Springer, Heidelberg, May / June 1998. doi: 10.1007/BFb0054134.

[53] M. J. Jacobson. Computing discrete logarithms in quadratic orders. *J. Cryptol.*, 13(4):473–492, jan 2000. ISSN 0933-2790. doi: 10.1007/s001450010013. URL `https://doi.org/10.1007/s001450010013`.

[54] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-Y. Kang. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

[55] M. Joye and B. Libert. A scalable scheme for privacy-preserving aggregation of time-series data. In A.-R. Sadeghi, editor, *FC 2013: 17th International Conference on Financial Cryptography and Data Security*, volume 7859 of *Lecture Notes in Computer Science*, pages 111–125. Springer, Heidelberg, Apr. 2013. doi: 10.1007/978-3-642-39884-1_10.

[56] S. Kadhe, N. Rajaraman, O. O. Koyluoglu, and K. Ramchandran. Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning. *CoRR*, abs/2009.11248, 2020. URL `https://arxiv.org/abs/2009.11248`.

[57] P. Kairouz, S. Oh, and P. Viswanath. Extremal mechanisms for local differential privacy. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL `https://proceedings.neurips.cc/paper_files/paper/2014/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf`.

[58] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1-2):1–210, 2021.

[59] A. Kate, E. V. Mangipudi, P. Mukherjee, H. Saleem, and S. A. K. Thyagarajan. Non-interactive vss using class groups and application to dkg. Cryptology ePrint Archive, Paper 2023/451, 2023. URL `https://eprint.iacr.org/2023/451`. `https://eprint.iacr.org/2023/451`.

[60] T. Kleinjung. Quadratic sieving. *Math. Comput.*, 85(300):1861–1873, 2016. doi: 10.1090/mcom/3058. URL `https://doi.org/10.1090/mcom/3058`.

[61] R. W. F. Lai and G. Malavolta. Subvector commitments with application to succinct arguments. In A. Boldyreva and D. Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 530–560. Springer, Heidelberg, Aug. 2019. doi: 10.1007/978-3-030-26948-7_19.

[62] I. Leontiadis, K. Elkhiyaoui, and R. Molva. Private and dynamic time-series data aggregation with trust relaxation. In D. Gritzalis, A. Kiayias, and I. G. Askoxylakis, editors, *CANS 14: 13th International Conference on Cryptology and Network Security*, volume 8813 of *Lecture Notes in Computer Science*, pages 305–320. Springer, Heidelberg, Oct. 2014. doi: 10.1007/978-3-319-12280-9_20.

[63] H. Li, H. Lin, A. Polychroniadou, and S. Tessaro. LERNA: Secure single-server aggregation via key-homomorphic masking. In J. Guo and R. Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part I*, volume 14438 of *Lecture Notes in Computer Science*, pages 302–334. Springer, Heidelberg, Dec. 2023. doi: 10.1007/978-981-99-8721-4_10.

[64] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling. Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019. ISBN 978-1-57735-809-1. doi: 10.1609/aaai.v33i01.33011544. URL `https://doi.org/10.1609/aaai.v33i01.33011544`.

[65] H. Lipmaa. Secure accumulators from euclidean rings without trusted setup. In F. Bao, P. Samarati, and J. Zhou, editors, *ACNS 12: 10th International Conference on Applied Cryptography and Network Security*, volume 7341 of *Lecture Notes in Computer Science*, pages 224–240. Springer, Heidelberg, June 2012. doi: 10.1007/978-3-642-31284-7_14.

[66] Z. Liu, S. Chen, J. Ye, J. Fan, H. Li, and X. Li. SASH: efficient secure aggregation based on SHPRG for federated learning. In J. Cussens and K. Zhang, editors, *Uncertainty in Artificial Intelligence, Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence, UAI 2022, 1-5 August 2022, Eindhoven, The Netherlands*, volume 180 of *Proceedings of Machine Learning Research*, pages 1243–1252. PMLR, 2022. URL `https://proceedings.mlr.press/v180/liu22c.html`.

[67] H. Lycklama, L. Burkhalter, A. Viand, N. Küchler, and A. Hithnawi. RoFL: Robustness of secure federated learning. In *2023 IEEE Symposium on Security and Privacy*, pages 453–476. IEEE Computer Society Press, May 2023. doi: 10.1109/SP46215.2023.10179400.

[68] Y. Ma, J. Woods, S. Angel, A. Polychroniadou, and T. Rabin. Flamingo: Multi-round single-server secure aggregation with applications to private federated learning. In *2023 IEEE Symposium on Security and Privacy*, pages 477–496. IEEE Computer Society Press, May 2023. doi: 10.1109/SP46215.2023.10179434.

[69] K. McCurley. Cryptographic key distribution and computation in class groups. *Proceedings of NATO ASI Number Theory and applications*, pages 459–479, 1989. URL `https://cir.nii.ac.jp/crid/1570854174866004864`.

[70] P. Mohassel and Y. Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (SP)*, pages 19–38. IEEE, 2017.

[71] M. Nasr, R. Shokri, and A. Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy*, pages 739–753. IEEE Computer Society Press, May 2019. doi: 10.1109/SP.2019.00065.

[72] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor. Federated learning for covid-19 detection with generative adversarial networks in edge cloud computing. *IEEE Internet of Things Journal*, 2021.

[73] J. Nguyen, K. Malik, H. Zhan, A. Yousefpour, M. Rabbat, M. Malek, and D. Huba. Federated learning with buffered asynchronous aggregation. In G. Camps-Valls, F. J. R. Ruiz, and I. Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 3581–3607. PMLR, 28–30 Mar 2022. URL `https://proceedings.mlr.press/v151/nguyen22b.html`.

[74] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, Heidelberg, Aug. 1992. doi: 10.1007/3-540-46766-1_9.

[75] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2018. doi: 10.1109/TIFS.2017.2787987.

[76] A. D. Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi. Calibrating probability with undersampling for unbalanced classification. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 159–166, 2015. doi: 10.1109/SSCI.2015.33.

[77] I. A. Seres, P. Burcsi, and P. Kutas. How (not) to hash into class groups of imaginary quadratic fields? Cryptology ePrint Archive, Paper 2024/034, 2024. URL `https://eprint.iacr.org/2024/034`. `https://eprint.iacr.org/2024/034`.

[78] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, nov 1979. ISSN 0001-0782. doi: 10.1145/359168.359176. URL `https://doi.org/10.1145/359168.359176`.

[79] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[80] J. Shao, Y. Sun, S. Li, and J. Zhang. Dres-fl: dropout-resilient secure federated learning for non-iid clients via secret data sharing. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9781713871088.

[81] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song. Privacy-preserving aggregation of time-series data. In *ISOC Network and Distributed System Security Symposium – NDSS 2011*. The Internet Society, Feb. 2011.

[82] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy*, pages 3–18. IEEE Computer Society Press, May 2017. doi: 10.1109/SP.2017.41.

[83] J. So, R. E. Ali, B. Güler, and A. S. Avestimehr. Secure aggregation for buffered asynchronous federated learning. *CoRR*, abs/2110.02177, 2021. URL `https://arxiv.org/abs/2110.02177`.

[84] J. So, R. E. Ali, B. Güler, and A. S. Avestimehr. Secure aggregation for buffered asynchronous federated learning, 2021.

[85] J. So, B. Guler, and A. S. Avestimehr. Codedprivateml: A fast and privacy-preserving framework for distributed machine learning. *IEEE Journal on Selected Areas in Information Theory*, 2(1):441–451, 2021.

[86] J. So, B. Güler, and A. S. Avestimehr. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. *IEEE Journal on Selected Areas in Information Theory*, 2(1):479–489, 2021. doi: 10.1109/JSAIT.2021.3054610.

[87] 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, 2022.

[88] S. A. K. Thyagarajan, G. Castagnos, F. Laguillaumie, and G. Malavolta. Efficient CCA timed commitments in class groups. In G. Vigna and E. Shi, editors, *ACM CCS 2021: 28th Conference on Computer and Communications Security*, pages 2663–2684. ACM Press, Nov. 2021. doi: 10.1145/3460120.3484773.

[89] I. Tucker. *Functional encryption and distributed signatures based on projective hash functions, the benefit of class groups*. Theses, Université de Lyon, Oct. 2020. URL `https://theses.hal.science/tel-03021689`.

[90] M. van Dijk, N. V. Nguyen, T. N. Nguyen, L. M. Nguyen, Q. Tran-Dinh, and P. H. Nguyen. Asynchronous federated learning with reduced number of rounds and with differential privacy from less aggregated gaussian noise. *arXiv preprint arXiv:2007.09208*, 2020.

[91] B. Wesolowski. Efficient verifiable delay functions. In Y. Ishai and V. Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 379–407. Springer, Heidelberg, May 2019. doi: 10.1007/978-3-030-17659-4_13.

[92] B. Wesolowski. Efficient verifiable delay functions. *Journal of Cryptology*, 33(4):2113–2147, Oct. 2020. doi: 10.1007/s00145-020-09364-x.

[93] C. Xie, S. Koyejo, and I. Gupta. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934*, 2019.

[94] C. Yang, J. So, C. He, S. Li, Q. Yu, and S. Avestimehr. Lightsecagg: Rethinking secure aggregation in federated learning. *CoRR*, abs/2109.14236, 2021. URL `https://arxiv.org/abs/2109.14236`.

[95] T. H. Yuen, H. Cui, and X. Xie. Compact zero-knowledge proofs for threshold ECDSA with trustless setup. In J. Garay, editor, *PKC 2021: 24th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 12710 of *Lecture Notes in Computer Science*, pages 481–511. Springer, Heidelberg, May 2021. doi: 10.1007/978-3-030-75245-3_18.

[96] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu. BatchCrypt: Efficient homomorphic encryption for Cross-Silo federated learning. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pages 493–506. USENIX Association, July 2020. ISBN 978-1-939133-14-4. URL `https://www.usenix.org/conference/atc20/presentation/zhang-chengliang`.

[97] L. Zhang, B. Shen, A. Barnawi, Y. Tang, Z. Luo, W. Wang, W. Zhang, and Z. Han. Feddpgan: federated differentially private generative adversarial networks framework for the detection of covid-19 pneumonia. *Information Systems Frontiers*, 23(6):1403–1415, 2021.

[98] Y. Zhao and H. Sun. Information theoretic secure aggregation with user dropouts. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 1124–1129, 2021. doi: 10.1109/ISIT45174.2021.9517953.

Table 3: Comparison of the RSA Modulus Size and the Class Group Size for various Security Levels. All the sizes are in bits. The RSA Modulus forms the basis of the DCR-based construction.

Security Level	RSA Size Modulus	Class Group Size
112	2048	1348
128	3072	1872
192	7680	3598
256	15360	5971

A Deferred Discussion on Class Groups and Cryptography

A.1 Class Group and Cryptography

Class Group-based Cryptography has its roots in the late 1980s, with the pioneering work of [18] and [69]. They proposed that the class group of ideals of maximal orders of imaginary quadratic fields might offer better security compared to the multiplicative group of finite fields. One of the key features of class group cryptography is its suitability for protocols involving multiple parties while requiring only a one-time transparent (or public-coin) setup without extensive interaction. This has enabled the construction of verifiable random functions without a trusted setup by [91, 92], accumulators by [65, 15], encryption switching protocols by [26], designated verifier non-interactive zero knowledge proofs of knowledge by [31], SNARKs [61, 20, 5], homomorphic secret sharing, and pseudorandom correlation functions for oblivious transfer by [3], range proofs by [34, 35], and vector commitments by[6]. Subsequent advancements in the computation of the structure of class groups of quadratic imaginary number fields were made by [50], and further improvements were presented in the works of [53],[11], and [60]. However, the computational costs of these algorithms increase with the size of Δ, with the largest current computation involving 512 bits for the size of Δ by [10]. It is worth noting that the subexponential complexity of computing class groups is asymptotically slower than integer factorization. Indeed, in the work of [12], it was conjectured that one needs the size of the discriminant of class group Δ_k to be 1872 bits to achieve 128 bits of security. Meanwhile, one needs an RSA modulus of 3072 bits to achieve 128-bit security. Some additional parameters for our class groups are: the size of $\bar{s}$ is 914 bits and the size of the prime p is 128 bits. We present the comparison in Table 3 that details the sizes for various levels of security. Looking ahead, the presence of the RSA modulus and related cryptographic assumptions is the main reason why BatchCrypt [96] and the works of [55] and [62] have inefficient parameter sizes, when compared to class group based protocol.

Efforts to improve the efficiency of cryptosystems based on class groups were undertaken by Hühnlein *et al.* [52]. However, their work was later attacked by [23] and [25]. Despite this setback, there has been a notable resurgence in the use of class groups in various applications over the past decade. [24] made a significant contribution by designing a cryptosystem based on a subgroup of a class group where discrete logarithms are easy to compute. This framework has since become the foundation of several protocols, including projective hash functions used to construct inner-product functional encryption [27], two-party and fully-threshold ECDSA signatures [28, 29, 95, 36], coin-mixing [47], and secure timed commitments [88]. Additionally, the framework has been employed to build secure multiparty computation from threshold encryption [17] and non-interactive verifiable secret sharing [59, 22]. We rely on this framework for our protocol.

A.2 Expanded Discussion on CL Framework

Broadly, the framework is defined by two functions - $\mathsf{CLGen}, \mathsf{CLSolve}$ with the former outputting a tuple of public parameters. The elements of this framework are the following:

- *Input Parameters*: κ_c is the computational security parameter, κ_s is a statistical security parameter, a prime p such that $p > 2^{\kappa_c}$, and uniform randomness ρ that is used by the CLGen algorithm and is made public.
- *Groups*: $\widehat{\mathbb{G}}$ is a finite multiplicative abelian group, $\mathbb{G}$ is a cyclic subgroup of $\widehat{\mathbb{G}}$, $\mathbb{F}$ is a subgroup of $\mathbb{G}$, $\mathbb{H} = \{x^p, x \in \mathbb{G}\}$
- *Orders*: $\mathbb{F}$ has order p, $\widehat{\mathbb{G}}$ has order $p \cdot \hat{s}$, $\mathbb{G}$ has order $p \cdot s$ such that s divides $\hat{s}$ and $gcd(p, \hat{s}) = 1, gcd(p, s) = 1$, $\mathbb{H}$ has order s and therefore $\mathbb{G} = \mathbb{F} \times \mathbb{H}$.
- *Generators*: f is the generator of $\mathbb{F}$, g is the generator of $\mathbb{G}$, and h is the generator of $\mathbb{H}$ with the property that $g = f \cdot h$
- *Upper Bound*: Only an upper bound $\bar{s}$ of $\hat{s}$ (and s) is provided.
- *Additional Properties*: Only encodings of $\widehat{\mathbb{G}}$ can be recognized as valid encodings and $s, \hat{s}$ are unknown.

- *Distributions*: $\mathcal{D}$ (resp. $\mathcal{D}_p$) be a distribution over the set of integers such that the distribution $\{g^x : x \leftarrow\!\!\$\ \mathcal{D}\}$ (resp. $\{g_p^x : x \leftarrow\!\!\$\ \mathcal{D}_p\}$) is at most distance $2^{-\kappa_s}$ from the uniform distribution over $\mathbb{G}$ (resp. $\mathbb{H}$).

Remark 3. The motivations behind these additional distributions are as follows. One can efficiently recognize valid encodings of elements in $\widehat{\mathbb{G}}$ but not $\mathbb{G}$. Therefore, a malicious adversary $\mathcal{A}$ can run our constructions by inputting elements belonging to $\widehat{\mathbb{G}}^p$ (rather than in $\mathbb{H}$). Unfortunately, this malicious behavior cannot be detected which allows $\mathcal{A}$ to obtain information on the sampled exponents modulo $\bar{\omega}$ (the group exponent of $\widehat{\mathbb{G}}^p$). By requiring the statistical closeness of the induced distribution to uniform in the aforementioned groups allows flexibility in proofs. Note that the assumptions do not depend on the choice of these two distributions. Further, the order s of $\mathbb{H}$ and group exponent $\bar{\omega}$ of $\widehat{\mathbb{G}}^p$ are unknown and the upper bound $\bar{s}$ is used to instantiate the aforementioned distribution.

We also have the following lemma from Castagnos, Imbert, and Laguillaumie [26] which defines how to sample from a discrete Gaussian distribution.

Lemma 1. *Let $\mathbb{G}$ be a cyclic group of order n, generated by g. Consider the random variable X sampled uniformly from $\mathbb{G}$; as such it satisfies $\Pr[X = h] = \frac{1}{n}$ for all $h \in \mathbb{G}$. Now consider the random variable Y with values in $\mathbb{G}$ as follows: draw y from the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z},\sigma}$ with $\sigma \geqslant n\sqrt{\frac{\ln(2(1+1/\epsilon))}{\pi}}$ and set $Y := g^y$. Then, it holds that:*

$$\Delta(X, Y) \leqslant 2\epsilon$$

Definition 2 (Class Group Framework)**.** *The framework is defined by two algorithms* $(\mathsf{CLGen}, \mathsf{CLSolve})$ *such that:*

- $\mathsf{pp} = (p, \kappa_c, \kappa_s, \bar{s}, f, h, \widehat{\mathbb{G}}, \mathbb{F}, \mathcal{D}_G, \mathcal{D}_H, \rho) \leftarrow\!\!\$\ \mathsf{CLGen}(1^{\kappa_c}, 1^{\kappa_s}, p; \rho)$
- *The* DL *problem is easy in $\mathbb{F}$, i.e., there exists a deterministic polynomial algorithm* $\mathsf{CLSolve}$ *that solves the discrete logarithm problem in $\mathbb{F}$:*

$$\Pr\left[x = x' \;\middle|\; \begin{array}{l} \mathsf{pp} = \leftarrow\!\!\$\ \mathsf{CLGen}(1^{\kappa_c}, 1^{\kappa_s}, p; \rho) \\ x \leftarrow\!\!\$\ \mathbb{Z}/p\mathbb{Z}, X = f^x; \\ x' \leftarrow \mathsf{CLSolve}(\mathsf{pp}, X) \end{array} \right] = 1$$

A.3 Why CL Framework?

Cryptographic protocols are often proven secure, under a hardness assumption. The hardness assumption guarantees that an adversary cannot break this assumption in polynomial time. A common setting for these cryptographic protocols is cyclic groups, usually of prime order q Let G be such a group and because it is cyclic, there exists a generator g such that $G := \{g^0, \ldots, g^{q-1}\}$. Now, consider the simpler setting where we want to securely sum up integer values. The immediate question is how to we encrypt these integer values when working over cyclic groups. The solution is to take your input x and map it to an element in the group G as g^x. This was the idea behind a famous encryption scheme known as ElGamal encryption proposed by Taher ElGamal [39]. Unfortunately, the security of this encryption scheme also required that given a random group element $g' \in G$, one cannot efficiently recover x' such that $g' = g^{x'}$, i.e., that the discrete logarithm of g' cannot be efficiently computed. For our use case of the server recovering the sum of the integers, using such an encoding scheme would require the server to compute the discrete logarithm which is inefficient. However, while computing the discrete logarithm for a random element is inefficient, the problem becomes simpler if we assume that the maximum value of the discrete logarithm is bounded. In other words, rather than searching over the entire set $\{0, \ldots, q-1\}$, the problem is simplified to searching over $\{0, \ldots, B-1\}$, where $B << q$. This is the assumption made by Shi et al. [81] and Guo et al. [49].

The CL Framework also works over cyclic groups but its security does not rely on the inefficiency of computing the discrete logarithm. Instead, it relies on the inefficiency of computing the order of the group. This gives us the ability to encode input x in the subgroup F where discrete logarithm is efficient while using the element in $\mathbb{H}$ to mask the inputs. This gives us the ciphertext $ct := h^k \cdot f^x$ where k is some random key. Note that if the order of $\mathbb{H}$ (call it s) is known, then the security is lost. This is because I can compute $ct^s = h^{ks} \cdot f^{sx}$ with h^{ks} being the identity element. Then, sx can be recovered because the discrete logarithm is efficiently computable, with x recoverable after. Therefore, the benefit of the CL framework is the following: Encode the elements into a cyclic group, from which the sum can be recovered efficiently without restricting the input sizes.

B Experimental Details

ABIDES Framework. To simulate evaluate real network conditions, we run the ABIDES simulator [21]. ABIDES supports a latency model which consists a base delay and plus a jitter that controls the percentage of messages that arrive within a given time. We set the base delay to the "global" setting in ABIDES's default parameters (the range is 21 microseconds to 53 milliseconds), and use the default parameters for the jitter.

Table 4: Details about Datasets

	Fraud Detection	MNIST	CIFAR-10	CIFAR-100
No. of classes	2	10	10	100
No. of training samples	$\tilde{2}$13k	60,000	50,000	50,000
No. of test samples	$\tilde{7}$1k	10,000	10,000	
Feature Details	30 features	Image Size: 28×28	Image Size: 32×32	Image Size: 32×32
Quantization Multiplier	10^4	2^{16}	2^{16}	2^{16}
License	Open Database License	Creative Commons Attribution-Share Alike 3.0 License	Apache License 2.0	MIT License

Dataset Details. The license and feature details of our datasets can be found in Table 4.

Architecture Details for MLP Classifier. The default neural network defined by sklearn.neural_network.MLPClassifier has the following characteristics. Note that our goal is to show that the secure aggregation does not impact accuracy when compared to learning in the clear.

- Architecture:
  - Multi-layer perceptron (MLP) with one hidden layer
  - Hidden layer contains 100 neurons by default
- Activation function:
  - ReLU (Rectified Linear Unit) for the hidden layer
  - Softmax for the output layer
- Optimizer:
  - Adam (Adaptive Moment Estimation)
- Learning rate:
  - Initial learning rate set to 0.001
  - Uses adaptive learning rate ('constant' schedule)
- Regularization:
  - L2 regularization with alpha=0.0001
- Batch size:
  - Mini-batch gradient descent with a batch size of 200
- Maximum iterations:
  - Default maximum number of iterations is 200
- Early stopping:
  - Not enabled by default
- Initialization:
  - Xavier initialization for weight initialization

Microbenchmarking Robust-PICASO. For completeness, we implement Robust-PICASO on the BICYCL library [16] to show its running time. For completeness, we also benchmark the running time of the encryption and aggregation protocols steps of Robust-PICASO in Figure 4.

C Deferred Proofs

Theorem 2. *Under the* DDH $- f$ *assumption (see Definition 1), Algorithm 2 is secure in the random oracle model.*

The proof of security of this theorem is through a sequence of hybrids. As mentioned in the introduction, the adversary triggers a challenge phase by presenting a set of honest clients whose inputs it wants to recover: $H_1, \ldots, H_t$ while also presenting two sets of challenge inputs for these clients $x_1, \ldots, x_t$ and $x'_1, \ldots, x'_t$. The challenger chooses to encrypt one of these sets at random and provide it to the attacker. While we defer the formal proof due to space constraints, we present a description of the hybrids below:

- Hybrid 0: The challenger outputs 1 if the adversary guesses correctly which of the challenge set was chosen, else outputs 0.

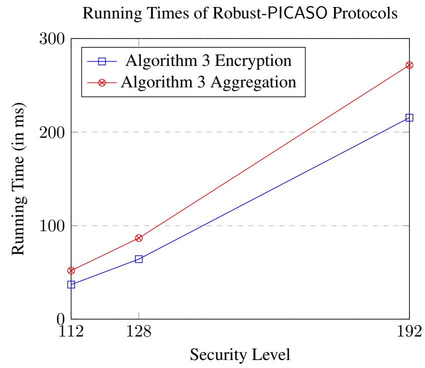


Figure 4: Performance of Robust-PICASO. Here, security level indicates the computational power needed to break the security of the protocol.

- Hybrid 1: The change is that for every query to the random oracle $\mathcal{H}$, the challenger tosses a biased coin δ_t. The biasing of the coin is that it takes 1 with probability $1/q_{enc} + 1$ and 0 with the remaining probability. Then let's define an event E: if for target iteration τ, $\delta_tau = 0$ *or* for some $t \neq \tau$, for an honest user i, there was an encryption query which produced $\delta_t = 1$. If E happens, then challenger just outputs a random bit. $\Pr[\neg E] = \frac{1}{q_{enc}+1} \cdot \left(\frac{q_{enc}}{q_{enc}+1}\right)^{q_{enc}} \geqslant \frac{1}{e(q_{enc}+1)}$
- Hybrid 2: Now, further modify the generation of $\mathcal{H}(t)$. If $\delta_t = 0$, then we sample $w_t \leftarrow\!\!\$\, \mathcal{D}_H$ and setting $\mathcal{H}(t) := h^{w_t}$. Meanwhile, if $\delta_t = 1$, it additionally samples $u_t \leftarrow\!\!\$\, \mathbb{Z}/p\mathbb{Z}$ setting $\mathcal{H}(t) := h^{w_t} \cdot f^{u_t}$. Note that the view between Hybrids 1 and 2 only happens when $\delta_t = 1$ and this is protected by the DDH-f assumption.

Note that if E has not happened, then the challenge epoch has $\delta_\tau = 1$, which implies that $\mathcal{H}(\tau) := h^{w_\tau} \cdot f^{u_\tau}$. In other words, every $\mathsf{ct}_{i,\tau}$ will be represented as: $f^{x_{i,\tau}+u_\tau} \cdot \mathsf{mask}_{i,\tau}$. We will then show that this u_τ provides sufficient masking of the inputs. Observe that for the honest parties, the server does not know any information about k_i. Meanwhile, $\mathsf{pk}_{i,\tau}$ is also not provided to the adversary. Therefore, at the challenge epoch, we sample individual values $u_{i,\tau}$ for honest i. Each of these $u_{i,\tau}$ masks the inputs $x_{i,\tau}$. Meanwhile, to ensure correctness of decryption, the challenger correctly simulates AUX_τ.

D Other Extensions to PICASO

D.1 Asynchronous Secure Aggregation

While FL algorithms have handled the problem of stragglers by simply considering them as dropouts. However, Asynchronous FL [30, 32, 90, 93] which focused on updating the global model, as soon as the local updates are received, even if the updates are for an old iteration. This was designed to ensure that stragglers in an iteration do not delay the global model update. Increased staleness in local models leads to greater errors when updating the global model [93], which was remedied by their proposed staleness-aware weighted averaging protocol called FedAsync. Unfortunately, Asynchronous FL is not easily composable with existing Secure Aggregation techniques. [73] presented an approach where the local updates are buffered, before updating the global model, with the buffer being stored in a trusted execution environment (TEE) to guarantee privacy. TEEs are known to be expensive. Later, BASecAgg[84] avoided the use of TEE by composing the buffering technique with a secure aggregation protocol, with the aggregation incorporating the staleness function. Unfortunately, BASecAgg required that each client share its mask with every other client, for every iteration which is undesirable for asynchrony.

BASecAgg. BASecAgg [84] successfully combined techniques of secure aggregation with asynchronous FL. Specifically, the server aggregates the model weights as: $\sum_i \phi(t - t_i) \cdot x_i$ where t_i is the iteration count of client i, with the corresponding updates being x_i. $\phi(t - t_i)$ is a "staleness" function which is 1 if $t = t_i$ and is monotonically decreasing. BASecAgg presented a solution where the aggregation of shares accounted for this staleness function. This is because LightSecAgg [87], unlike other works[13, 8], had the server reconstruct the sum of the masks masks for the online clients which can then be used to unmask. Similarly, PICASO also only helps the server reconstruct the online clients' masks. Therefore, using PICASO as the secure aggregation component of BASecAgg, we can build an asynchronous secure aggregation protocol that avoids the expensive secret sharing costs associated with LightSecAgg.

D.2 Minimizing Trust Assumption

Existing protocols that offer secure aggregation rely on a non-colluding assumption. Typically, this is modeled by allowing the server to collude with up to a threshold t, out of n parties, with the security being completely lost if even $t + 1$ parties are corrupted. It follows that if the server colludes with all n parties, then privacy is lost. These parties are those involved in helping reconstruct the sum, even in the presence of dropouts. In SecAgg, these parties are fellow clients while in LERNA and Flamingo, this corresponds to the committee members.

However, PICASO can be easily extended to support a committee of M collectors. In this setting, the server can corrupt to a certain threshold t of collectors and need the help of at least $1 + t$ of collectors to reconstruct the sum.

A naive implementation would be to secret-share the current public key among the committee of such collectors. This technique is called secret sharing [78]. More recently, Braun et al. [17] demonstrated how to construct secret sharing techniques, compatible with the CL Framework. However, it is to be noted that this comes at a significant cost:

- The communication cost and computation cost scales linearly with the size of the committee.
- The server's computation and communication costs also increase. Additionally, the server has to engage in expensive coordination with the committee. Factoring in networking delays, it is entirely possible that different committee members receive inputs from different subsets of clients. Now, the server has to find the intersection of online clients among this list.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and precede the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper sets out to solve a critical problem in prior work on secure aggregation. In this work, we demonstrate how to reduce the synchronization by employing one additional party. We demonstrate experiments to show competitive performance over prior work. In addition, we also train machine learning models to justify that our protocol can be used for its intended purpose.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have a conclusion paragraph that draws attention to some of the limitations while identifying how they can be remedied in future work.

   Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper introduces all the necessary theoretical framework and assumptions for security of the construction. There's detailed proof deferred to the appendix.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The protocols are well detailed, including the parameter settings for our classifier. We use publicly available ABIDES framework to simulate real-life networking situations. For our class group operations, we take the BICYCL framework that is open-source and bind it to Python.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.

- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [NA]

   Justification: Unfortunately, there was no support for supplementary material upload. However, we are happy to furnish the anonymized code for interested reviewers.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We detail the quantization metrics along with the choice of datasets with the test-train splits. Our choice of training algorithms are vanilla versions, with no customized hyperparameters.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [NA]

   Justification: Our running time experiments report the mean of 30 iterations, for various choices of number of clients. Meanwhile, we present the accuracy values, as a function of iterations for various datasets. At each iteration, the data is randomly split.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We detail the system settings of the device on which experiments are performed.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The paper complies with the code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The work focuses on privacy of client-held data. This is surveyed in the introduction and motivates why privacy-preserving federated learning is important and its positive impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We do not use any of the stated models or data sources.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The experiments have only used publicly available datasets with their license details specified in a tabular column.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We are not releasing any new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: There were no human subjects involved in this project.

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We do not have any research with human subjects that forms a part of this work.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.