ACCESS-FL: Agile Communication and Computation for Efficient Secure Aggregation in Stable Networks for FLaaS

Niousha Nazemi[1], Omid Tavallaie[2], Shuaijun Chen[1], Anna Maria Mandalari[3], Kanchana Thilakarathna[1], Ralph Holz[4], Hamed Haddadi[5], Albert Y. Zomaya[1]
[1]School of Computer Science, The University of Sydney, Australia
[2]Department of Engineering Science, University of Oxford, United Kingdom
[3]Electronic & Electrical Engineering Department, University College London (UCL), United Kingdom
[4]Department of Computer Science, University of Münster, Germany
[5]Department of Computing, Imperial College London, United Kingdom
{niousha.nazemi, shuaijun.chen, kanchana.thilakarathna, albert.zomaya}@sydney.edu.au,
{omid.tavallaie}@eng.ox.ac.uk, {a.mandalari}@ucl.ac.uk, {ralph.holz}@uni-muenster.de, {h.haddadi}@imperial.ac.uk

***Abstract*—Federated Learning (FL) enables privacy-preserving machine learning by allowing clients to collaboratively train models without sharing raw data. Federated Learning as a Service (FLaaS) extends this approach to cloud infrastructures. However, conventional secure aggregation protocols, such as Google's SecAgg and SecAgg+, introduce high computation and communication overheads, particularly in large-scale FLaaS deployments where client dropout rates are limited. To address these challenges, we propose ACCESS-FL, a lightweight, secure aggregation method designed for honest-but-curious FLaaS scenarios with stable network conditions. ACCESS-FL eliminates double masking, Shamir's Secret Sharing, and excessive encryption/decryption by creating shared secrets only between two peers per client, which reduces computation and communication complexity to constant $O(1)$ and makes the algorithm independent of network size and comparable to standard FL. ACCESS-FL preserves privacy against inversion attacks and maintains model accuracy equivalent to the FL, SecAgg, and SecAgg+ protocols, proving that reducing overhead does not compromise learning performance and achieves communication and computation costs comparable to standard FL. Experimental evaluations on benchmark datasets (MNIST, FMNIST, and CIFAR-10) demonstrate lower overhead, making ACCESS-FL practical for service-based stable FLaaS applications such as healthcare analytics.**

***Index Terms*—Federated Learning as a Service (FLaaS), secure aggregation, communication cost, computation cost, honest-but-curious scenario.**

I. Introduction

Federated Learning (FL) [1] is a promising approach to privacy-preserving collaborative learning that enables clients to keep data local while exchanging model updates with a central aggregator. *Federated Learning as a Service (FLaaS)* [2] has extended this concept to managed cloud infrastructures [3], [4]. However, FL remains vulnerable to *model inversion* attacks [5] when the aggregator is *honest-but-curious* [6], [7], and can reconstruct sensitive client data by analyzing model updates. To mitigate such privacy risks, Google proposed *Secure Aggregation (SecAgg)* [8], and *SecAgg+* [9], by masking client updates using cryptographic techniques. However,

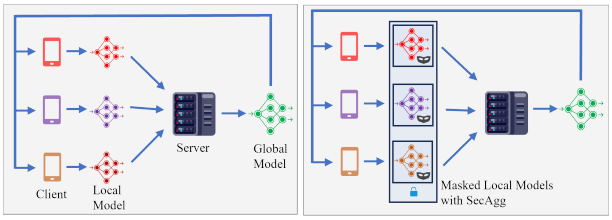


Fig. 1: Comparison between vanilla FL and FL with SecAgg.

Google's protocols lead to $O(|C|^2)$ and $O(|C|\log|C|)$ overhead, respectively, where $|C|$ is the number of clients. These overheads become especially higher in large-scale FLaaS scenarios where the number of clients can be substantial, but the network is relatively stable (i.e., low dropout rates). Fig. 1 illustrates the SecAgg approach compared to traditional FL.

In this paper, we propose *ACCESS-FL*, a *first-of-its-kind* lightweight secure aggregation protocol (to the best of our knowledge) that achieves *constant overhead* $O(1)$ per client in stable FLaaS environments with the honest-but-curious aggregator, such as fraud detection [10] and the IBM anti-money-laundering system [11]. Additionally, healthcare providers deploy FLaaS with platforms such as NVIDIA Clara [12], where a cloud-side coordinator manages encrypted model updates while each hospital keeps imaging data private [13], [14]. In the aforementioned applications, privacy is crucial as participants and the aggregator are deployed and reliable elements of a system. Instead of creating shared secrets with all other clients (SecAgg) or $\log(|C|) - 1$ neighbors (SecAgg+), each client in ACCESS-FL derives *only two shared secrets* (generated based on Diffie-Hellman (DH) [15] algorithm) for masking. Additionally, ACCESS-FL eliminates private masking, Shamir's Secret Sharing (SSS) [16], [17], and all encryption/decryption, reducing both communication and computation (caused by state-of-the-art protocols) to constant $O(1)$ per client while preserving privacy in honest-but-curious scenarios. This cost makes ACCESS-FL more efficient for

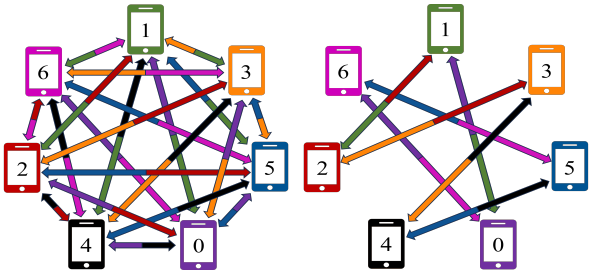


Fig. 2: SecAgg (left) vs. ACCESS-FL (right) in pair selection for shared masking.

large-scale, enterprise-grade FLaaS *stable* networks where dropout rates and adversarial activities are limited, including multi-hospital collaborations. The main contributions of this paper are as follows:

1) $O(1)$ overhead: We propose, to the best of our knowledge, the first secure aggregation protocol specified for *stable* FL networks that achieve $O(1)$ communication and computation overhead per client, independent of network scale.

2) Dynamic, privacy-preserving pairing: We introduce a dynamic client pairing mechanism (based on a deterministic function) with a secret seed (accessible only to clients), to hide the exact pairs from the aggregator (Fig. 2).

3) FLaaS Efficiency: By generating just two shared secrets per client and removing the need for SSS, encryption/decryption, and private masking, we significantly simplify the secure aggregation workflow in FLaaS.

4) Experimental validation: We demonstrate on multiple benchmark datasets (MNIST [18], Fashion-MNIST [19] and CIFAR10 [20]) that ACCESS-FL preserves the same model accuracy as vanilla FL while reducing overhead and runtime compared to SecAgg and SecAgg+.

5) Open-source code: This is publicly available at [21].

The rest of this paper is organized as follows: Section II explains the state-of-the-art protocols (SecAgg and SecAgg+). Section III explains security primitives. Section IV reviews related work. ACCESS-FL is explained in Section V. Section VI evaluates ACCESS-FL's performance in comparison to state-of-the-art protocols. We conclude the paper and discuss future directions in Section VII.

II. Preliminary Study: SecAgg and SecAgg+

In this section, we provide an overview of Google's Secure Aggregation protocol (SecAgg) and its improved variant (SecAgg+). Table I summarizes notations used in this paper.

A. SecAgg

SecAgg proceeds in eight rounds as follows: (1) *Model broadcast:* The server broadcasts the initial global model to clients. (2) *Key pair generation:* Client i generates *two* private-public key pairs as (SK_i^1, PK_i^1) and (SK_i^2, PK_i^2). Then, it sends its public keys to the server. (3) *Broadcasting public keys:* The server broadcasts public keys to all clients. (4) *Secret splitting & encryption:* Client $i \in C$ generates a random seed b_i and splits b_i and SK_i^1 into $|C|$ parts (using SSS algorithm).

TABLE I: Declaration of main notations

Notations	*Definition*
n	Round number of FL
C	The list of participating clients in an FL round
$\|C\|$	Number of clients in the FL system
$\|D_i\|$	Size of client i's local dataset
SK_i	Private key of client i
PK_i	Public key of client i
PK_{all}	List of public keys
$s_{i,j}$	Shared secret generated by SK_i and PK_j
fp_i	First pair of client i for generating shared secret
sp_i	Second pair of client i for generating shared secret
$m_{i,j}$	Shared mask between client i and client j
w_i	Local model parameters trained by Client i
w_i^{msk}	Masked model update sent by client i to the server
G	Global model

Then, it encrypts each part for peer $j \in C$ (by the key derived from SK_i^2 and PK_j^2) to generate $b_{i,j}$ and $SK_{i,j}^1$. Finally, it sends the ciphertext $e_{i,j} = (i||j||b_{i,j}||SK_{i,j}^1)$ to the server. (5) *Ciphertext distribution:* The server creates the participants set as C_1, then forwards the $(|C|-1)$ ciphertexts designed for each client along with C_1 set to the correspondent client. (6) *Masking:* Client i creates $(|C|-1)$ shared secrets with every client j as $s_{i,j}$ (by using SK_i^1 and PK_j^1), then expands b_i and $s_{i,j}$ by the pseudo-random generator function PRG to create a private mask m_i and shared masks $m_{i,j}$ $\forall j \in C_1$, respectively. Finally, the client applies these masks to its local model w_i and computes its masked model w_i^{mask}, which is sent to the server. (7) *Participant reveal:* The server announces the set C_2 of clients that sent their masked models. Then, each active client i decrypts ciphertext $e_{j,i}$ $\forall j \in C_1$ (by using a key generated from SK_i^2 and PK_j^2) to obtain $b_{j,i}$ $\forall j \in C_2$ of the participants and $m_{j,i}$ $\forall j \in C_1 \setminus C_2$ of the dropped-out clients, and returns them to the server. (8) *Global model aggregation:* The server reconstructs random seed of participants and shared secrets of dropped-out clients (by SSS algorithm), then expands each reconstructed value by PRG to generate private mask m_j $\forall j \in C_2$ and shared masks $m_{j,i}$ $\forall j \in C_1 \setminus C_2$. Finally, it aggregates the global model by:

$$\sum_{i \in \{C_2\}} w_i^{msk} - \sum_{i \in \{C_2\}} m_i + \sum_{i \in C_2, j \in \{C_1 \setminus C_2\}} m_{j,i}. \quad (1)$$

In SecAgg, the communication cost for each client and the server are calculated as $O(|C|)$ and $O(|C|^2)$, respectively [8].

B. SecAgg+

SecAgg+ [9] is an improvement over SecAgg designed to reduce computational and communication costs. Instead of generating shared secrets with every client, the server generates a random k-regular graph for $k = (\log |C|)$, where $|C|$ is the number of clients. Clients only generate shared masks with their neighbors in this graph. Although SecAgg+ reduces the costs compared to SecAgg, for a larger number of clients, it leads to unnecessary overhead in stable networks.

III. Security Primitives

This section introduces the fundamental cryptography used in ACCESS-FL, including the pseudo-random generator function and the key agreement protocol:

Algorithm 1: Client i's algorithm in ACCESS-FL to generate a key pair (in the first training round).

Waits for the server to send the initial G; # Algorithm 4.
Waits for a trusted third party to send public parameters;
Generates a key pair with public parameters.
$(SK_i, PK_i) \leftarrow key\_gen(param)$;
Stores SK_i securely;
Sends PK_i to the server;
Waits to receive PK_{all} from the server; # Algorithm 4.

Algorithm 2: Client i's algorithm in ACCESS-FL to find two pairs (during all training rounds).

Calculates the $distance$ value to find its pairs
$set_i^n = \{d \mid d \in [1, \lfloor(|C|-1)/2\rfloor], d \neq distance_i^{n-1}\}$;
$distance_i^n = RandInt(set_i^n)$;
Finds its pairs from the sorted participant list.
$fp_i \leftarrow (i + distance_i^n)\%|C|$; # 1^{st} pair idx
$sp_i \leftarrow (i - distance_i^n + |C|)\%|C|$; # 2^{nd} pair idx
Generates shared masks with peers by Algorithm 3.

A. The Pseudo-Random Generator function (PRG) [22], [23] is a deterministic function that produces a sequence of outputs that appear random from a given seed input. In this paper, PRGs are used to generate masks out of secrets.

B. Key Agreement Protocols [24], such as Diffie-Hellman (DH) [25] and Elliptic Curve Diffie-Hellman (ECDH) [26], enable two parties to derive the same secret without exposing it to the server [27] through the following steps: *(1) Public parameters:* A trusted third party generates public parameters by $param \leftarrow param_gen(key_size)$function. These parameters include a large prime number p and a generator g modulo p and are shared between both clients. *(2) Key pairs:* Each client generates a public-private key pair through $(SK, PK) \leftarrow key\_gen(param)$ function and shares its PK with its peers. Despite using the same $param$, the key pairs generated by both clients are unique to each individual. *(3) Shared secret:* Each client computes the shared secret using its SK and the peer's PK. Both clients arrive at the same shared secret value, which seeds the PRG for pairwise masking.

IV. Related Work

Various studies have focused on secure aggregation for FL to optimize communication and computation. CESA [28] theoretically reduces communication cost. Authors in [29] proposed a non-interactive key establishment protocol, removing Shamir's Secret Sharing to reduce overhead. FastSecAgg [30] used a multi-secret sharing approach [31] to lower computation costs while keeping communication costs similar to SecAgg. Addressing communication overhead, the SAFER method [32] compressed neural network updates by using TopBinary Coding and one-bit quantization. Additionally, SAFELearn [33] with only two communication rounds per iteration, supports client dropouts and avoids trusted third parties. Furthermore, Rathee et al. [34] introduced ELSA to reduce communication and computation costs. Authors in [35] addressed honest-but-curious scenarios. The FAlkor protocol proposed in [36] employs GPU acceleration and stream cipher-based masking for computational efficiency. Moreover, SHFL [37], EdgeSA [38], and BSDA [39] proposed secure FL approaches in edge environments. Finally, authors in [40]–[43] analyzed FL secure aggregation to identify security issues. Nevertheless, in the mentioned schemes, communication and computation still increase with the number of clients, especially in stable FLaaS. ACCESS-FL removes this growth by proposing constant-cost secure aggregation in honest-but-curious scenarios.

Algorithm 3: Client i's algorithm in ACCESS-FL to generate its masked model (during all training rounds).

Generates two shared secrets with its two pairs.
$s_{i,fp_i} \leftarrow key\_agree(SK_i, PK_{fp_i})$; # 1^{st} shared secret.
$s_{i,sp_i} \leftarrow key\_agree(SK_i, PK_{sp_i})$; # 2^{nd} shared secret.
Creates its shared masks by PRG function.
$m_{i,fp_i} \leftarrow PRG(s_{i,fp_i})$; # 1^{st} shared mask.
$m_{i,sp_i} \leftarrow PRG(s_{i,sp_i})$; # 2^{nd} shared mask.
Determines signs based on indices
if $fp_i < i$ *then* $sign_{fp} \leftarrow -1$ *else* 1;
if $sp_i < i$ *then* $sign_{sp} \leftarrow -1$ *else* 1;
Calculates its masked model.
$w_i^{msk} \leftarrow w_i + sign_{fp} \times m_{i,fp_i} + sign_{sp} \times m_{i,sp_i}$;
Sends w_i^{msk} to the server;
Waits to receive the new G from the server; # Algorithm 4.

Algorithm 4: Server's algorithm in ACCESS-FL.

First training round
Broadcasts initial G;
Waits for all clients to send their public keys; # Algorithm 1.
for $\forall i \in C$ **do**
  $PK_{all} \cup [PK_i]$; # List of public keys
end
Broadcasts list of PK_{all};
All training rounds
Waits for clients to send their masked models; # Algorithm 2.
if $|\{ i \in C \mid w_i^{msk} \}| < |C|$ **then**
  # Updates C with participants.
  for $\forall i \in C$ **do**
    Sends updated C to client i;
  end
  Waits for clients to send their new w^{msk}; # Algorithm 5.
end
$G \leftarrow 0$;
$G \leftarrow \frac{1}{|D|} \sum_{i \in C} |D_i| \; w_i^{\text{msk}}$; # Aggregates masked models
Broadcasts new G;

V. ACCESS-FL Protocol

This section proposes ACCESS-FL, a communication and computation-efficient secure aggregation protocol for *stable* FLaaS deployments with limited client dropouts and low delay variations in honest-but-curious scenarios, as follows:

1) Initialization: The server broadcasts the initial global model. Each client receives common public parameters from a *trusted third party* to generate a unique DH public-private key pair *once for all FL rounds* and sends the public key to the server (Algorithm 1). Then, *the server broadcasts public keys once in ACCESS-FL for all FL rounds*. Unlike SecAgg and SecAgg+, no new keys are generated per round; therefore, the key setup

Fig. 3: Finding new pairs in the presence of a client dropout.

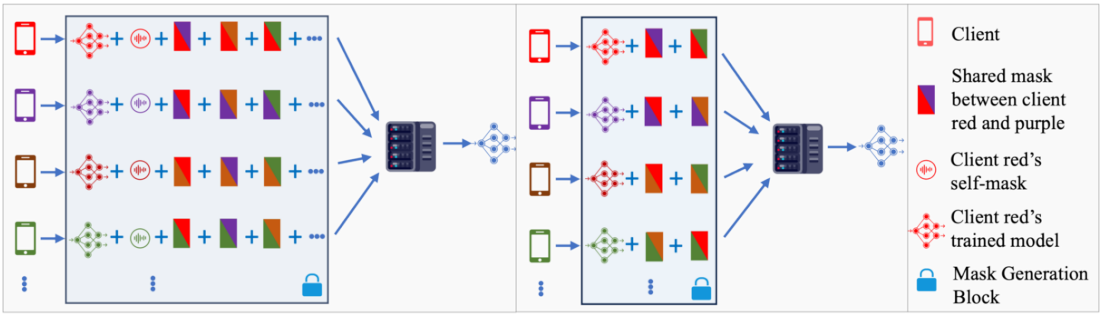


Fig. 4: Comparison between SecAgg and ACCESS-FL.

Algorithm 5: Client i's algorithm in ACCESS-FL for handling client dropout or delayed updates.

```
# Calculates the new_distance_i^n.
set_i^n = {d | d ∈ [1, ⌊(|C| − 1)/2⌋], d ≠ distance_i^{n−1}, d ≠ distance_i^n};
new_distance_i^n = RandInt(set_i^n);
distance_i^n ← new_distance_i^n;
Finds new pairs with new distance by Algorithm 2;
```

traffic drops from $(2 \times |C|)$ keys/round to $(|C|)$ keys in total, thereby costs associated with key pair generation and distribution are significantly reduced in ACCESS-FL.

2) Pairs Selection: Clients are sorted into a participating list, and each runs *a trusted deterministic function* (only accessible to clients), given the current training round number and the public parameters as the seeds. This function generates a random integer value, called as distance, within the range $[1, \lfloor(|C|-1)/2\rfloor]$. *The distance generator function leads to the same value for all clients and varies at each training round.* Then each client i pairs with client $((i + distance) \mod |C|)$ and client $((i + distance) \mod |C|)$ (where $|C|$ is the number of clients in the sorted participating list), to generate *two shared secrets* (each shared secret is derived by the client's private key and the peer's public key). This dynamic pair selection process ensures that clients have different pairing partners in each round. To prevent identical pairs in different rounds, the function excluded the distances used in previous rounds. The mentioned process is detailed in Algorithm 2.

3) Shared Masks Generation: For each pair (i, j), clients derive an identical shared secret s via DH ($s_{i,j} = s_{j,i}$), feed it to the PRG to generate shared mask ($m_{i,j} = m_{j,i}$), and set opposite signs based on each client's index in the participant list, that is, the one with a smaller index gets the -1 coefficient to make sure shared masks cancel out each other in the aggregation process without server intervention (unlike SecAgg and SecAgg+). Thus, $m_{i,j} + m_{j,i} = 0$.

4) Local Training & Masking: Each client trains the global model on its local dataset (*ACCESS-FL is designed to be independent of model and data distribution type*) and adds the two pairwise masks to its update. The masked model is then sent to the server; hence, the server does not have access to the plain-trained model of each client (Algorithm 3).

5) Global Model Update: If all masked models arrive on time, the server aggregates them to generate the new global model (Algorithm 4 illustrates the process of ACCESS-FL running at the server), otherwise it broadcasts the participant list, and then every active client *recomputes the distance, forms new pairs, and resends a freshly masked model* (Algorithm 5). This step is mandatory to remove shared masks of dropped-out clients without server involvement (unlike SecAgg and SecAgg+) and keeps the aggregated model identical to vanilla FL.

ACCESS-FL is designed for stable FLaaS environments with limited client dropouts and low delay variations in honest-but-curious scenarios. Hence, the server does not get stuck in the same training round waiting for new masked models. Fig. 3 illustrates the masked model recomputation process when one of eight clients drops out. Fig. 4 contrasts the complexity

and message volume in one FL round between SecAgg (left) and ACCESS-FL (right). In SecAgg, every client obscures its update with *one private mask* plus $(|C|-1)$ pairwise masks. Thus, mask generation and message count grow with the number of clients. ACCESS-FL keeps the model-privacy goal with the reduced cost: each client derives masks with only *two* peers, a total independent of $(|C|)$. The server then sums the masked updates into the new global model. Fewer masks mean fewer messages and a single masking layer, which delivers the same privacy with significantly lower communication and computation costs.

A. Message Passing in ACCESS-FL

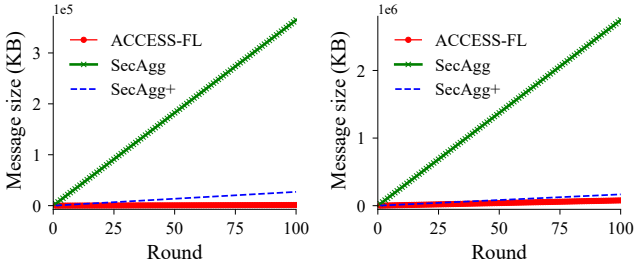
This section analyzes the total number of messages exchanged between the server and C clients over n training FL rounds in ACCESS-FL. This process is categorized into three phases as follows: *(1) Initialization*: The server broadcasts the initial model. Each client sends a single public key to the server $(|C|)$. The server, then broadcasts the set of all public keys. *(2) Shared mask generation at first training round*: Client i finds two pairs as $fp_i = [(i + distance) \mod |C|]$ and $sp_i = [(i - distance + |C|) \mod |C|]$. Here, $distance$ is a random integer within the range of $[1, \lfloor(|C|-1)/2\rfloor]$ (where $|C| \geq 6$). Having $distance$ more than $(|C|-1)/2$ makes the chosen pair equal to previously found pairs. After finding the pairs, each client generates two shared secrets and expands them with PRG to create two shared masks (denoted as m_{i,fp_i}) and m_{i,sp_i}. *(3) All training rounds*: Each round, client i sends one masked update to the server as $w_i^{msk} = w_i + m_{i,fp_i} + m_{i,sp_i}$, where w is the trained model. Then, the server generates the new global model by aggregating all masked models. Since $m_{i,fp_i} = -m_{fp_i,sp_{fp_i}}$, all masks cancel out each other, and the sum of masked models equals the sum of unmasked trained models. Finally, the server broadcasts the new global model G to all clients. Across n FL rounds, clients send $(n+1) \times |C|$, and the server sends $n+1$ messages. Thus, both client and server cost remain $O(1)$.

VI. Evaluation Results

We benchmark ACCESS-FL, SecAgg and SecAgg+ on three image datasets (MNIST, Fashion-MNIST, and CIFAR-10) using a 2-layer neural network (2NN) and a convolutional neural network (CNN). All runs last 100 FL rounds and are optimized with SGD (learning rate = 0.01). To assess the communication and computation costs, we present results based on the accumulated message size, the number of exchanged messages, and the running time for both clients and the server in ACCESS-FL, SecAgg, and SecAgg+.

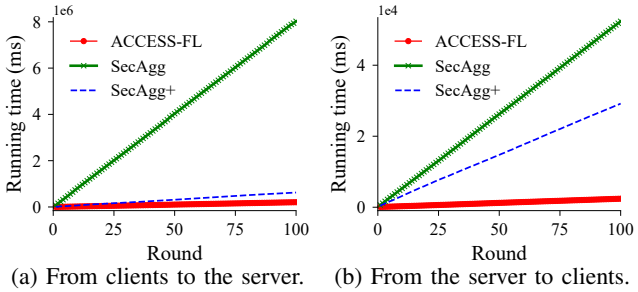
A. Communication Cost of ACCESS-FL, SecAgg and SecAgg+

Figs. 5a and 5b illustrate the accumulated message size sent from clients to the server and from the server to clients, respectively, for ACCESS-FL, SecAgg, and SecAgg+ over 100 rounds. We observe that the total size of the message for each client in ACCESS-FL remains constant through the 100 rounds. Furthermore, *the communication cost for each client does not increase with the number of participating clients*, since each client only generates shared masks with two pairs. In contrast, in SecAgg, every client pairs with all others, and in SecAgg+, with $log(|C|)-1$ neighbors, thus their communication cost increases with the network size, reaching over 350 MB and 30 MB after 100 rounds, respectively. The server's accumulated message size is also affected by the protocol design, that is, only around 80 MB for ACCESS-FL versus 3000 MB (SecAgg) and 200 MB (SecAgg+) by the 100th round. The accumulated number of messages exchanged between the server and clients are demonstrated in Tables II and III. Each client in ACCESS-FL sends 101 messages (one public key plus 100 masked updates), and the server only *broadcasts public keys in the initial round and the aggregated model in each round*. Thus, the server overhead in large-scale stable FLaaS networks is essentially the same as in traditional FL. In contrast, each client in SecAgg and SecAgg+ sends around 20,000 and 1,700 messages, respectively, while the server sends about 10,000 and 700 messages, which include broadcasting participant list, global model, and two public keys per client, as well as sending $(|C|-1)$ ciphertexts per client per each round. Clients in ACCESS-FL *find their pairs and change them in every round without the server's knowledge*; the only per-round payload is the masked model update, whose size stays constant with network scale. Hence, *ACCESS-FL improves the privacy of an honest-but-curious stable FL system with approximately the same client load as traditional FL.* SecAgg and SecAgg+ have much higher communication costs: each client must create many pairwise secrets (all peers or $log|C|-1$ neighbors), generate two key pairs, and exchange encrypted values. The server in SecAgg and SecAgg+ learns the pairings and must manage global-model unmasking, whereas *in ACCESS-FL the server does not know the peers and is not responsible for unmasking.*



(a) From clients to the server. (b) From the server to clients.

Fig. 5: Accumulative message size (MNIST dataset).



(a) From clients to the server. (b) From the server to clients.

Fig. 6: Accumulated running time (MNIST dataset).

TABLE II: Total number of messages sent from clients in scenarios with *node dropout (D) and without node dropout (ND).*

Round	*ACCESS-FL (ND)*	*ACCESS-FL (D)*	*SecAgg (ND)*	*SecAgg (D)*	*SecAgg+ (ND)*	*SecAgg+ (D)*	*FedAvg (ND)*
10	1100	1099	202000	200802	17000	20999	1000
30	3100	3067	606000	590482	51000	62367	3000
50	5100	4995	1010000	964530	85000	102895	5000
70	7100	6883	1414000	1323266	119000	142583	7000
100	10100	9640	2020000	1833360	170000	200540	10000

TABLE III: Total number of messages sent from server in scenarios with *node dropout (D) and without node dropout (ND).*

Round	*ACCESS-FL (ND)*	*ACCESS-FL (D)*	*SecAgg (ND)*	*SecAgg (D)*	*SecAgg+ (ND)*	*SecAgg+ (D)*	*FedAvg (ND)*
10	12	13	100041	99041	7041	6033	11
30	32	35	300121	291201	21121	17915	31
50	52	57	500201	475601	35201	29557	51
70	72	79	700281	652401	49281	40959	71
100	102	112	1000401	903701	70401	57612	101

TABLE IV: Accuracy comparisons between *ACCESS-FL (AC)* and *FedAvg (FA)* for MNIST, FMNIST, and CIFAR10.

	MNIST		*FMNIST*		*CIFAR10*	
Round	*AC*	*FA*	*AC*	*FA*	*AC*	*FA*
10	0.5126	0.5126	0.4711	0.4711	0.1129	0.1137
30	0.6783	0.6783	0.6339	0.6338	0.1465	0.1477
50	0.7338	0.7338	0.6862	0.6862	0.2176	0.2117
70	0.7675	0.7675	0.7084	0.7084	0.2764	0.2720
100	0.7970	0.7970	0.7247	0.7247	0.3157	0.3109

These properties make ACCESS-FL an efficient solution for large-scale, stable FLaaS deployments, such as *healthcare systems where data privacy is crucial*. Fig. 6b illustrates the accumulated running time for the server in ACCESS-FL, SecAgg, and SecAgg+ over 100 training rounds. In ACCESS-FL, the server's running time at each round is essentially constant (except for the initial broadcast of public keys) since it *only* aggregates masked updates and is not responsible for unmasking or handling dropouts (with 100 clients, the per-round cost stays flat). In SecAgg, the curve rises sharply; SecAgg+ reduces that cost by half. The higher overhead in SecAgg and SecAgg+ arises from cryptographic operations needed to reconstruct shared masks for dropped-out clients and private masks for participants, which increases aggregation complexity. Although SecAgg+ reduces the overall cost by generating the random k-regular graph and having each client peered with $(log|C|) - 1$ neighbors instead of $|C| - 1$, it has a higher cost compared to ACCESS-FL.

Fig. 6a plots the accumulated client running time. For ACCESS-FL, the curve stays low and flat and never grows with $|C|$. Each client (1) runs the deterministic pairing function, (2) trains locally, (3) derives two pairwise masks (PRG on two shared secrets), and (4) uploads the masked update. Dropout handling is triggered *only when the server announces a missing client*. SecAgg+ reduces SecAgg's cost; however, they rise with $|C|$ since each client must create two key pairs, build $(log|C|-1)$ shared secrets, apply double masking, execute PRG once for the private mask and $(log|C| - 1)$ times for pairwise masks, run SSS, create ciphertext for every neighbor, and decrypt the peers' ciphertext; the server must then reconstruct masks for dropouts. Learning curves (Table IV) demonstrate that *ACCESS-FL and FedAvg are identical on MNIST and FMNIST and differ by <1 % on CIFAR-10*, showing ACCESS-FL's accuracy scales across datasets.

B. Client Dropout for ACCESS-FL, SecAgg, and SecAgg+

We evaluate ACCESS-FL, SecAgg, SecAgg$^+$ and traditional FL under a *10% dropout of 100 clients over 100 rounds* on MNIST. Tables II and III report message counts in dropout and non-dropout scenarios. In ACCESS-FL, total messages fall slightly, from 10,100 to 9,640 by round 100, since every dropout leads to one extra resend but also shrinks the participant set. In contrast, SecAgg and SecAgg+ retain their high overhead driven by pairwise secrets. ACCESS-FL shows only a minimal rise: the server's role (initial key broadcast and one model broadcast per round) is unchanged by dropouts. SecAgg and SecAgg+ remain heavy as the server must reconstruct shared masks. Thus, ACCESS-FL keeps communication almost as low as plain FL while still preserving privacy: each client needs *one public key* and *two shared masks*, *independent of* $|C|$. *To the best of our knowledge, this is the first secure aggregation scheme to achieve privacy with only two masks per client, with the cost of* $O(1)$*, in honest-but-curious scenarios.*

VII. Conclusion and Future Work

This paper proposed *ACCESS-FL*, an efficient, secure aggregation protocol for honest-but-curious scenarios in stable FLaaS deployments such as healthcare organizations. ACCESS-FL preserves the security of state-of-the-art protocols (SecAgg and SecAgg+); however, it forms shared secrets with *only two peers*, giving each client a *constant* $O(1)$ *cost*. By eliminating double masking, all encryption/decryption, and Shamir's Secret Sharing, ACCESS-FL reduces both communication and computation overhead. The proposed protocol also handles limited dropouts, with remaining participants finding new pairs for masking and resending masked updates with no server intervention or access to any mask (unlike SecAgg/SecAgg+). Experiments on MNIST, FMNIST, and CIFAR-10 demonstrated reductions in traffic needed for secure aggregation (lower message count and message volume) and running time, comparable to vanilla FedAvg while matching its accuracy and maintaining the privacy provided by SecAgg and SecAgg+ in honest-but-curious scenarios. Future work will extend ACCESS-FL to active-adversary settings, integrate differential privacy, and address loop vulnerabilities under frequent dropouts.

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, vol. 54, pp. 1273–1282, PMLR, 2017.

[2] N. Kourtellis, K. Katevas, and D. Perino, "Flaas: Federated learning as a service," in *Proceedings of the 1st workshop on distributed machine learning*, pp. 7–13, 2020.

[3] W. Gao, O. Tavallaie, S. Chen, and A. Zomaya, "Federated learning as a service for hierarchical edge networks with heterogeneous models," in *International Conference on Service-Oriented Computing*, pp. 85–99, Springer, 2024.

[4] S. Chen, O. Tavallaie, N. Nazemi, and A. Y. Zomaya, "RBLA: Rank-Based-LoRA-Aggregation for Fine-Tuning Heterogeneous Models in FLaaS," in *International Conference on Web Services*, pp. 47–62, Springer, 2024.

[5] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1322–1333, 2015.

[6] L. Kissner and D. Song, "Privacy-preserving set operations," in *Annual International Cryptology Conference*, pp. 241–257, Springer, 2005.

[7] L. E. Olson, M. J. Rosulek, and M. Winslett, "Harvesting credentials in trust negotiation as an honest-but-curious adversary," in *Proceedings of ACM workshop on Privacy in electronic society*, pp. 64–67, 2007.

[8] 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 *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

[9] 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*, CCS '20, (New York, NY, USA), p. 1253–1269, Association for Computing Machinery, 2020.

[10] W. Yang, Y. Zhang, K. Ye, L. Li, and C.-Z. Xu, "Ffd: A federated learning based method for credit card fraud detection," in *Big Data–BigData 2019: 8th International Congress, Held as Part of the Services Conference Federation, SCF 2019, San Diego, CA, USA, June 25–30, 2019, Proceedings 8*, pp. 18–32, Springer, 2019.

[11] N. Baracaldo, H. Shaul, N. Drucker, S. Kadhe, and H. Ludwig, "Building privacy-preserving federated learning to help fight financial crime," 2023. Accessed: 2024-06-01.

[12] "Nvidia clara." https://www.nvidia.com/clara/. Accessed: 2025-06-05.

[13] N. Rieke, J. Hancox, W. Li, F. Milletari, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, *et al.*, "The future of digital health with federated learning," *NPJ digital medicine*, vol. 3, no. 1, p. 119, 2020.

[14] A. Rahman, M. S. Hossain, G. Muhammad, D. Kundu, T. Debnath, M. Rahman, M. S. I. Khan, P. Tiwari, and S. S. Band, "Federated learning-based ai approaches in smart healthcare: concepts, taxonomies, challenges and open issues," *Cluster computing*, vol. 26, no. 4, pp. 2271–2311, 2023.

[15] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[16] E. Dawson and D. Donovan, "The breadth of shamir's secret-sharing scheme," *Computers & Security*, vol. 13, no. 1, pp. 69–78, 1994.

[17] L.-J. Pang and Y.-M. Wang, "A new (t, n) multi-secret sharing scheme based on shamir's secret sharing," *Applied Mathematics and Computation*, vol. 167, no. 2, pp. 840–848, 2005.

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[19] T. Datasets, "Fashion-mnist," 2024. Accessed: 2024-08-01.

[20] T. Datasets, "Cifar-10." https://www.tensorflow.org/datasets/catalog/cifar10.

[21] "Access-fl." https://github.com/SeeAccessFL/ACCESS-FL, 2024.

[22] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudo-random number generator," *SIAM Journal on computing*, vol. 15, no. 2, pp. 364–383, 1986.

[23] R. Impagliazzo, L. A. Levin, and M. Luby, "Pseudo-random generation from one-way functions," in *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pp. 12–24, 1989.

[24] S. Blake-Wilson, D. Johnson, and A. Menezes, "Key agreement protocols and their security analysis," in *IMA international conference on cryptography and coding*, pp. 30–45, Springer, 1997.

[25] W. Diffie and M. E. Hellman, "New directions in cryptography," in *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*, pp. 365–390, 2022.

[26] R. Haakegaard and J. Lang, "The elliptic curve diffie-hellman (ecdh)," *Online at https://koclab. cs. ucsb. edu/teaching/ecc/project/2015Projects/Haakegaard+ Lang. pdf*, 2015.

[27] F. Bao, R. H. Deng, and H. Zhu, "Variations of diffie-hellman problem," in *International conference on information and communications security*, pp. 301–312, Springer, 2003.

[28] N. Nazemi, O. Tavallaie, S. Chen, A. Y. Zomaya, and R. Holz, "Boosting communication efficiency of federated learning's secure aggregation," in *2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*, pp. 157–158, IEEE, 2024.

[29] K. Mandal, G. Gong, and C. Liu, "Nike-based fast privacy-preserving highdimensional data aggregation for mobile devices," *IEEE T Depend Secure*, pp. 142–149, 2018.

[30] S. Kadhe, N. Rajaraman, O. O. Koyluoglu, and K. Ramchandran, "Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning," *arXiv preprint arXiv:2009.11248*, 2020.

[31] P. Heckbert, "Fourier transforms and the fast fourier transform (fft) algorithm," *Computer Graphics*, vol. 2, no. 1995, pp. 15–463, 1995.

[32] C. Beguier and E. W. Tramel, "Safer: Sparse secure aggregation for federated learning," *arXiv preprint arXiv:2007.14861*, 2020.

[33] H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, H. Möllering, T. D. Nguyen, P. Rieger, A.-R. Sadeghi, T. Schneider, H. Yalame, *et al.*, "Safelearn: Secure aggregation for private federated learning," in *2021 IEEE Security and Privacy Workshops (SPW)*, pp. 56–62, IEEE, 2021.

[34] M. Rathee, C. Shen, S. Wagh, and R. A. Popa, "Elsa: Secure aggregation for federated learning with malicious actors," in *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 1961–1979, IEEE, 2023.

[35] Z. Liu, J. Guo, K.-Y. Lam, and J. Zhao, "Efficient dropout-resilient aggregation for privacy-preserving machine learning," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 1839–1854, 2022.

[36] M. Georgieva Belorgey, S. Dandjee, N. Gama, D. Jetchev, and D. Mikushin, "Falkor: Federated learning secure aggregation powered by aesctr gpu implementation," in *Proceedings of the 11th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pp. 11–22, 2023.

[37] O. Tavallaie, K. Thilakarathna, S. Seneviratne, A. Seneviratne, and A. Y. Zomaya, "Shfl: Secure hierarchical federated learning framework for edge networks," *International Conference on Service-Oriented Computing*, 2024.

[38] J. Bouamama, Y. Benkaouz, and M. Ouzzif, "Edgesa: Secure aggregation for privacy-preserving federated learning in edge computing," in *2023 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pp. 0375–0382, IEEE, 2023.

[39] X. Wang, S. Garg, H. Lin, G. Kaddoum, J. Hu, and M. S. Hossain, "A secure data aggregation strategy in edge computing and blockchain-empowered internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 14237–14246, 2020.

[40] M. Mansouri, M. Önen, W. B. Jaballah, and M. Conti, "Sok: Secure aggregation based on cryptographic schemes for federated learning," in *PETS 2023, 23rd Privacy Enhancing Technologies Symposium*, vol. 2023, pp. 140–157, 2023.

[41] J. Wu and W. Zhang, "On the security of verifiable and oblivious secure aggregation for privacy-preserving federated learning," *IEEE Transactions on Dependable and Secure Computing*, 2024.

[42] J. Wang, Z. Wang, and A. B. Abdallah, "Robust client selection based secure collaborative learning algorithm for pneumonia detection," in *2023 IEEE 6th International Conference on Knowledge Innovation and Invention (ICKII)*, pp. 614–619, IEEE, 2023.

[43] A. R. Elkordy, J. Zhang, Y. H. Ezzeldin, K. Psounis, and S. Avestimehr, "How much privacy does federated learning with secure aggregation guarantee?," *Proceedings on Privacy Enhancing Technologies*, vol. 1, pp. 510–526, 2023.