Secured Cost-effective Anonymous Federated Learning with Proxied Privacy Enhancement for Personal Devices

Muhammad S. Brennaf, Po Yang, Vitaveska Lanfranchi

***Abstract*—Privacy concerns have escalated due to companies' misuse of user data and the occurrence of data breaches and leaks worldwide. Uploading personal data from personal devices to a central server over the network poses a danger in obtaining an inference. Hence, a different approach is needed for this scenario. Federated learning enables collaborative training on devices while maintaining the privacy of user data. Federated learning originally aimed to address privacy concerns but is vulnerable to certain privacy attacks. Although certain privacy-enhancing strategies are available, researchers are actively seeking a more effective option. This research suggests two privacy improvement methods using proxies as a better option for personal devices in a federated learning environment, achieving good performance and cost effective without accuracy loss. We studied and assessed how the methodology compared to other methodologies. Finally, we discussed how this proposed technique can address the limitations of other techniques and possible collaborations with them.**

***Index Terms*—encryption, federated learning, privacy, proxy**

I. Introduction

ON-DEVICE training and inference have gained popularity as a prevailing trend, such as using IoT devices to obtain inference. Nevertheless, this pattern also gives rise to a research dilemma in which data stored on personal devices are often segregated due to many privacy concerns. Casually uploading a user's data for the purpose of making an inference can be subject to various risks, such as security breaches, violation of privacy regulations, or failure to ensure privacy protection for sensitive information. Alternatively, if we train an ML model on the device, it may result in erroneous inference because of its limited data and computational capability.

Federated learning, which was proposed by [1], offers a way out from these obstacles. By engaging in its collaborative learning, users can train together to generate a superior global model and attain a high level of accuracy. The utilisation of aggregation in federated learning empowers users to privately do machine learning tasks on their own gadgets, hence obviating the necessity of transmitting private information to a central server. Nevertheless, the technique is not completely secure from privacy attacks. [2] shows that a server employing a membership inference attack was able to determine the identity of a client by analysing their gradient changes.

Several privacy enhancement algorithms have been suggested to add privacy protection. These include 1) Differential privacy [3], which involves noises to protect personal information; 2) Secure aggregation [4], which allows to securely aggregate model parameters; and 3) Homomorphic encryption [5], which enables calculations to be performed on encrypted data. Nevertheless, there are certain limitations associated with them in specific scenarios, such as differential privacy's accuracy loss [6], secure aggregation's dropout concern [7], and homomorphic encryption's extensive computation [8], [9].

This work introduces a novel approach called anonymous federated learning with proxied privacy enhancement, which serves as an alternate method to enhance privacy in federated learning environments. This method allows for collaborative learning while maintaining privacy, as it ensures that the server does not gains knowledge about the model updates' source. This is particularly useful for personal devices with unique properties. The key contributions we have made include:

- We propose two novel proxy-based privacy-preserving ways that leverage proxy-based anonymisation to ensure privacy in anonymous federated learning. The methods enable providing strong security guarantees, low computational cost, and no impact on model accuracy, making them practical for resource-constrained IoT devices.
- We conduct an extensive empirical evaluation of robustness, scalability and efficiency of our proposed methods across multiple real-world datasets and FL scenarios. Our results demonstrate a significant reduction in communication overhead and memory usage—reducing client-side storage by up to 60%, while maintaining model accuracy.
- We analyse the interoperability of our methods with differential privacy, secure aggregation, and homomorphic encryption, showing how these methods can be combined to enhance security and usability in FL environments.

Our proposed approach offers a practical and deployable privacy-preserving solution for federated learning in IoT and edge computing contexts by balancing privacy, efficiency, and scalability. We also tackle key privacy risks in FL, such as collusion attacks, client tracing, and content integrity abuses. Our proposed solution effectively neutralises these threats, ensuring privacy protection while maintaining system performance.

II. Background Research

A. Federated Learning

Google introduced federated learning (FL) [1], [10], [11] as an innovative method for addressing issues related to data security and privacy. Federated learning is a decentralised

The authors are with the School of Computer Science, The University of Sheffield, Sheffield, S10 2TN UK e-mail: msbrennaf1@sheffield.ac.uk; po.yang@sheffield.ac.uk; v.lanfranchi@sheffield.ac.uk.

approach to ML where multiple users with distributed data work together to address an ML task [2], [12]. This framework operates in a collaborative manner, with the training and testing procedures taking place on the client's side rather than on the server's side. Initially, the server will select a subset of clients and allocate them a global model for a certain task. Subsequently, each client trains the model on their device by utilising their respective private records. Later, the client transmits their model updates, not the data, to the server. The server aggregates the incoming parameter updates and then generates a new global model. Following that, the process is iterated for another cycle until it achieves a specific threshold or fulfils the requirements, such as achieving high accuracy.

B. Further Privacy Protections

Essentially, federated learning is designed to tackle the privacy issues that users may have. However, the studies of [13] and [14] demonstrate that relying solely on federated learning does not sufficiently ensure protection against privacy breaches, namely membership inference attacks [2], [15]. Three commonly used privacy-preserving techniques are available to add more privacy guarantee to the FL users:

1) Differential Privacy: Proposed by [3], this method employs a contrasting approach to safeguarding data. While traditional methods aim to keep data completely confidential, differential privacy adopts a contrasting approach by permitting the disclosure of data with a certain level of added noise. The goal is to offer a general comprehension without specifying details: $Pr[A(D) \in S] \leq e^{\epsilon} Pr[A(D') \in S]$

[18] mentioned that conditions for differential privacy are when "ϵ-differential privacy is satisfied by a randomised procedure $A(D)$ when all datasets D and D', which vary by just a record, as well as all sets $S \in R$, which R is the A range and ϵ is a non-negative value that represents the privacy budget score." Thus, no particular information in the collection has a major effect on the algorithm's output distribution.

This strategy has numerous benefits, including the ability to facilitate sharing and establish a privacy budget. Nevertheless, in certain applications, such as healthcare, where accuracy and privacy are of utmost importance, differential privacy may not be the preferred approach due to the noised inference and accuracy drawback. Additionally, there is a potential for a stationary user base to experience data leakage [16], [17].

2) Secure Aggregation: The methodology proposed by [4] utilises Shamir's Secret Sharing [19] as its fundamental basis. The objective is to ensure that the server only has access to aggregated data rather than individual contributions. This technique allows clients to confidentially exchange a secret, such as a model update, with the aggregator. The aggregator, however, gains no knowledge other than the aggregate value. This technique becomes a favourable choice to consolidate model parameters securely in machine learning. However, the secret-sharing aspect of Secure Aggregation poses a scaling difficulty. An increase in the number of clients results in a corresponding increase in the level of complexity [20]. Power-limited equipment, including personal gadgets, might also face issues related to synchronisation and dropout [21]. Moreover, collusion might occur when the server conspires with some clients to obtain other clients' confidential information [7].

3) Homomorphic Encryption: The technique described by [5] intends to mitigate the risk of privacy breaches during processing of user data. Most of the confidential information are encrypted, but occasionally it needs to be decrypted to perform some operations, posing a potential privacy threat to the original data. Homomorphic encryption was specifically developed to enable the processing of data while maintaining its encrypted state, considered to be the 'Holy Grail' in cryptography [22]. Nonetheless, Fully Homomorphic Encryption is characterised by its high computing demands [23], resulting in processing speeds that are 1,000 to 1,000,000 times slower than similar plaintext operations and potentially as low as one operation per second [9]. Furthermore, calculation in a complex FL aggregation with distinct user public keys adds more challenge to it. Enabling FHE on portable and personal devices remains problematic due to the limited resources available and is currently regarded as unfeasible [8], [24].

C. Additional Privacy Guarantee Through Anonymity

Oblivious DNS over HTTPS (ODoH) [25] is a cutting-edge internet technology that aims to safeguard user privacy while maintaining optimal performance. Research conducted by [26] in Fig. 1 demonstrates that it exhibited strong performance in real-life simulations. In comparison to other privacy-focused DNS alternatives, the left image illustrates that ODoH reduces query time by fifty percent or more. The figure on the right indicates that ODoH consistently performs well under both normal and low-latency settings. These aspects are significant as privacy and performance seldom coexist harmoniously.

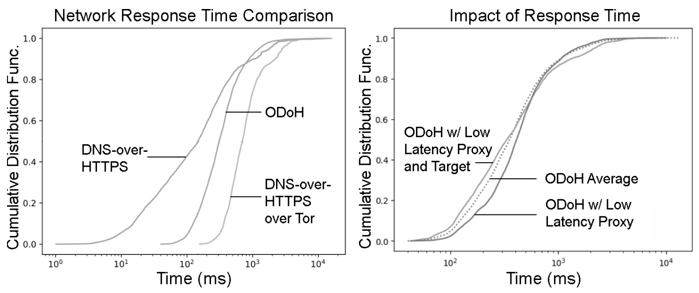


Fig. 1. Network response time (left) and impact (right) of ODoH [26]

Federated learning with proxy, drawing inspiration from ODoH, holds the potential to provide clients with enhanced anonymity and stronger privacy assurances. In contrast to earlier privacy enhancement methods, this method is anticipated to be devoid of any accuracy drawbacks, dropout problems, or computationally burdensome disadvantages. The primary function of the proxy is to obfuscate the sources' identity. This measure will mitigate adversary attacks, such as a reverse engineer attack, to disclose the origin through the content of the transmitted data. The underlying principle is that the server possesses knowledge of the needed data (i.e. model updates) but lacks awareness of the origin. Conversely, the proxy is aware of the source's identity but not the content. In this context, the content is concealed using an encryption method.

Several studies have been conducted on the implementation of anonymisation techniques in the field of ML. [25] devised a technique for obfuscating user identity during internet access by utilising a fortified intermediary server within an encrypted transaction. [27] presented a method for pseudonymisation in Recommendation-as-a-Service (RaaS) that involves using two layers of proxy. ProxyFL [28] facilitates the exchange of models via proxy in a decentralised FL system while also including differential privacy to enhance privacy. FedKAD by [29] offers an FL framework that utilises local Knowledge Aggregation on high-level feature maps and Knowledge Distillation.

III. Methodology

We utilise a reliable intermediary proxy in our experiments to facilitate anonymous federated learning. To bolster the safety of information and prevent the proxy from accessing the content, we utilise encryption techniques such as symmetric and asymmetric encryption. Symmetric encryption utilises one key for both the encryption and decryption processes. Conversely, asymmetric encryption utilises a pair of keys consisting of a public key and also a private one. Symmetric encryption is renowned for its superior speed and efficiency in computation, while asymmetric encryption is acknowledged for its heightened level of security and conveniency in distribution. Furthermore, a compression technique may be employed to reduce the communication cost, particularly for personal devices with restricted bandwidth. This work introduces two approaches: Two-Stage Communication (2SC), which follows a server-client-server pattern, and Three Stage Communication (3SC), which follows a client-server-client-server pattern.

A. Two-stage Communication (2SC)

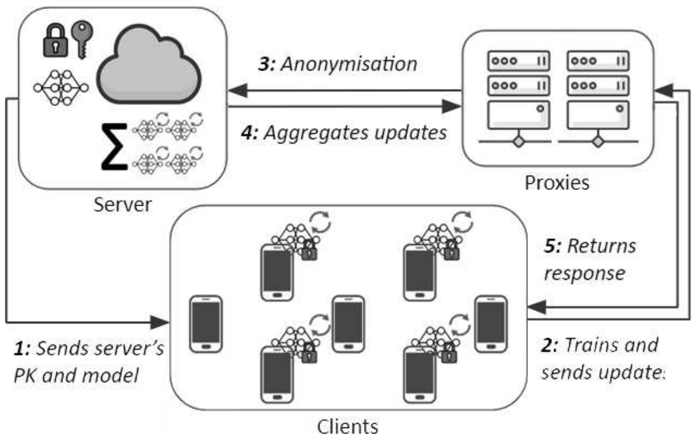


Fig. 2. Two-stage communication anonymised federated learning

Fig. 2 and Algorithm 1 describe Two-Stage Communication. The aggregate function is based on the original FedAvg algorithm [10]. In general, the server generates a public key (PK) and a secret key (SK). The server selects certain clients for the training round and sends the PK along with the global model to the clients. The client trains the model on their local device and sends their encrypted (i.e. using the given PK) model update to the server through an arbitrary proxy. The

Algorithm 1 `Two-Stage Communication.` The users are denoted by U, indexed by u, and anonymised into x. The server determines the selection of the aggregation function and the specifications for model training.

```
Run on the server:
  initialise $gm_0, PK$, and $SK$
  for each round $i$ = 1, 2, ... do
    $U_i \leftarrow$ (arbitrary set of clients)
    for each client $u \in U_i$ concurrently do
      $e^x_{i+1} \leftarrow \text{UserTrain}(u, gm_i, PK)$
      $gm^x_{i+1} \leftarrow \texttt{decrypt}(e^x_{i+1}, SK)$
    end for
    $gm_{i+1} \leftarrow$ run an aggregate function for all $gm^x_{i+1}$
  end for

UserTrain$(u, gm, PK)$: // Run on client $u$
  $gm \leftarrow$ run a model training locally
  $e \leftarrow \texttt{encrypt}(gm, PK)$
  $P \leftarrow$ (an arbitrary proxy)
  $S \leftarrow$ (the server's address)
  ProxyForward$(P, e, S)$

ProxyForward$(P, e, S)$: // Run on proxy $P$
  remove source's identity
  forward $e$ to $S$
```

proxy then strips the source's identity before forwarding the content to the destination. Subsequently, the server aggregates the decrypted (i.e. using the SK) incoming model update to create a new global model and be ready for the next round.

Users are accountable for choosing a reliable or arbitrary proxy to minimise the chances of collusion between the proxy and server. Validation of each request is required by both the server and proxy. Subsequently, a response is expected from them to indicate the success or failure of a request. In the beginning of the process, the server can send the global model and their public key directly to the clients for various reasons:

1) *Redundancy*: The server does not require the necessity to hide their identity from the clients.
2) *Reduce collusion risk*: Selecting specific proxies by the server can heighten the likelihood of collusion between them and possibly compromise the user's identity.
3) *Prevent adversary attacks*: Proxy's unawareness of the server's public key serves as a safeguard against any manipulation or falsification to the transmitted data.

B. Three-stage Communication (3SC)

Unlike the 2SC approach, the 3SC scheme initiates with a request from voluntary clients to the server, explained by Fig. 3 and algorithm 2. The server first creates a training round key pair (PK_s and SK_s) and shares the public key PK_s with all clients. Eligible clients can choose whether or not to participate in the training session and the volunteers create a key pair (PK_u and SK_u). The client then sends their encrypted (using the server's PK_s) public key (PK_u) to the server through an arbitrary proxy. The proxy removes the

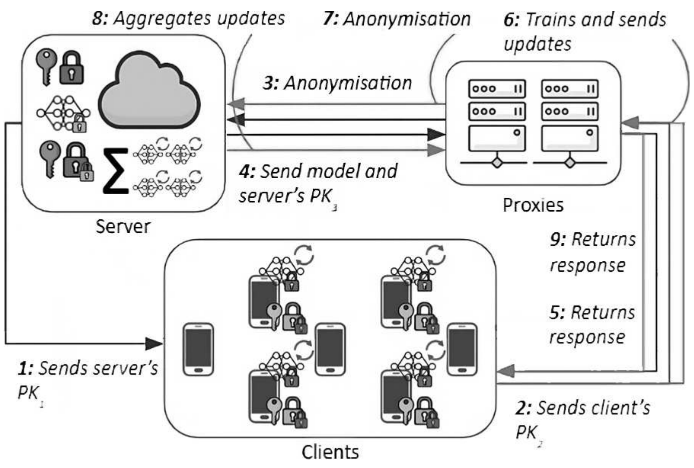


Fig. 3. Three-stage communication anonymised federated learning

Algorithm 2 `Three-Stage Communication.` The users are denoted by U, indexed by u, and anonymised into x. The server determines the selection of the aggregation function and the specifications for model training.

```
Run on the server:
  initialise $gm_0, PK_s$, and $SK_s$
  for each round $i$ = 1, 2, ... do
    initialise $PK_i$, and $SK_i$
    $U_i \leftarrow$ (arbitrary set of voluntary clients)
    for each client $u \in U_i$ concurrently do
      $ePK_u \leftarrow$ RequestModel$(u, PK_s)$
      $PK_u \leftarrow \texttt{decrypt}(ePK_u, SK_s)$
      $eGM_i^u \leftarrow \texttt{encrypt}(gm_i, PK_u)$
      $e_{i+1}^x \leftarrow$ UserTrain$(u, eGM_i^u, PK_i)$
      $gm_{i+1}^x \leftarrow \texttt{decrypt}(e_{i+1}^x, SK_i)$
    end for
    $gm_{i+1} \leftarrow$ run an aggregate function for all $gm_{i+1}^x$
  end for

RequestModel$(u, PK_s)$: // Run on client $u$
  initialise $PK_u$ and $SK_u$
  $e \leftarrow \texttt{encrypt}(PK_u, PK_s)$
  $P \leftarrow$ (an arbitrary proxy)
  $S \leftarrow$ (the server's address)
  ProxyForward$(P, e, S)$

UserTrain$(u, eGM_u, PK_i)$: // Run on client $u$
  $gm \leftarrow \texttt{decrypt}(eGM_u, SK_u)$
  $gm_2 \leftarrow$ run a model training locally using $gm$
  $e \leftarrow \texttt{encrypt}(gm_2, PK_i)$
  $P \leftarrow$ (an arbitrary proxy)
  $S \leftarrow$ (the server's address)
  ProxyForward$(P, e, S)$

ProxyForward$(P, e, S)$: // Run on proxy $P$
  remove source's identity
  forward $e$ to $S$
```

client's identity before sending the content to the target. Once receiving it, the server creates a model training key pair (PK_i and SK_i), then send the PK_i and current global model in encrypted form (using client's PK_u) back to the client through the proxy. Using SK_u, the client decrypt the response, train the global model on their device, and send the encrypted (using server's PK_i) model update to the server through a trusted proxy. Subsequently, the steps are similar to the 2SC.

While the 2SC provides simpler, faster, and less expense in computation and communication, the later offers more security through stronger encryption measures and can withstand against dropout, participation abuse, and client collusion.

IV. Experiments

Several experiments are carried out to understand how the methodology performs in different test scenarios. Each experiment will be evaluated according to the purpose, primarily in computation and communication efficiency to suit personal restricted device conditions. While accuracy is measured, the goal is not to achieve high accuracy—rather to show there is no accuracy drawback compared to the basic FL. We do not do a thorough examination of memory utilisation, as our objective is to gain a broader insight of implementation on constrained personal devices. The whole test report can be accessed through: https://s.id/ieeeiot.

A. Interoperability and Compatibility Test

This set of experiments aims to ensure compatibility and interoperability of the methodology across different scenarios: running in PC, in mobile device, with CPU only, with GPU, and comparison between vanilla (basic) FL, proxied FL with symmetric encryption, and proxied FL with asymmetric encryption. The tests are simulated using instances of modern web browsers, which have extensive compatibility across various platform and sufficient access to utilise the machine's resources, such as memory, sensors, and GPU [42]. The proxy and server will operate on a PC, while the clients will operate on a smartphone and a PC, representing personal devices.

In our experiments, we utilise TensorFlow.js[1] and DISCO.js [30] for the learning platform backbone, FedAVG [10] for the aggregation algorithm, and an optimised CNN model by Chris's [31] with MNIST from Kaggle dataset[1] for the training rounds. Symmetric AES and asymmetric key pair[1] are selected for encryption scenarios. We employ around 10 mainstream web browsers, including Chrome, Edge, Firefox, as well as their mobile version (e.g. Chrome Mobile), and their developer version (e.g. Firefox Nightly), added with Samsung Mobile. Testing on mobile phone involves a smaller set of data compared to the PC in order to generate faster results.

Based on the results shown in Tables I and II, the accuracies are mainly consistent and preserved due to no data modification nor noise introduced. No package lost was recorded (package lost results in decryption failure). The amount of time required for all mode operations is generally similar, except on GPU tests. Moreover, the encrypteds are slightly 2-6% slower but cost 1.5 to 2.4 in bandwidth compared to the basic FL.

[1]https://gist.github.com/senoyodha/9e964ec18155ebecef14ea21a539430f

TABLE I
VANILLA AND PROXIED FEDERATED LEARNING ON PC

PC	Trn.	Val.	Test	Time	Comm.
Vanilla (CPU)	97.9%	99.5%	99.6%	657s	236.2MB
Vanilla (+GPU)	97.7%	99.2%	99.6%	294s	236.1MB
AES (CPU)	98.1%	99.4%	89.9%	676s	472.4MB
AES (+GPU)	97.6%	99.3%	99.6%	330s	472.4MB
Key Pair (CPU)	98.0%	99.2%	99.6%	683s	557.8MB
Key Pair (+GPU)	97.5%	99.1%	99.6%	456s	557.8MB

TABLE II
VANILLA AND PROXIED FEDERATED LEARNING ON MOBILE DEVICE

Mobile	Trn.	Val.	Test	Time	Comm.
Vanilla	94.7%	99.2%	95.2%	1,375s	75.0MB
AES	96.5%	99.3%	98.6%	1,441s	149.9MB
Key Pair	96.8%	99.3%	98.9%	1,431s	177.0MB

In summary, excluding communication costs, utilising an asymmetric key pair would be the optimal choice for a PC for a superior security. Conversely, it would be more suitable to employ AES or less complex encoding method for bandwidth limited mobile cases, while maintaining its privacy. Alternatively, employing more robust encryption methods along with compression can provide enhanced security and efficiency.

Remarkably, we were unable to deploy the widely used RSA encryption due to multiple constraints. Crypto.js[1] is primarily intended for server-side NodeJS usage rather than being compatible with client-side browsers. As a result, the function to produce the key pair cannot be accessed from the client side. Additionally, employing the Web Crypto API[2] is recommended for cryptographic implementation in this scenario, yet, its functionality is limited to a secure environment. Due to a local host being utilised, the installation of trustworthy SSL certificates is not feasible. Lastly, the gradient update message surpasses the maximum permissible message size. While the gradient update size is around 1.88MB, the maximum allowable message size is 190 bytes in RSA 2048 with SHA-256 encoding [33], which is 10,000 times larger.

B. Compression Against Encryption Test

Encryptions applied provides heightened security, yet, the data size would increase due to the transformation. This set of tests aims to reduce the sent data size in order to save user's bandwidth by applying a compression layer. The assessment will be evaluated according to its data size and speed.

Several compression algorithms are compared including LZ, Pako, FFlate, UZIP, and LZMA with all possible setups (e.g. string, deflate, gzip, zlib, and encoding compressions) [34]. As for encryption, we utilise Crypto's symmetric AES and TweetNaCl's asymmetric PK (based on the Poly1305 one-time authenticator with the XSalsa20 stream cypher)[1]. Similar to earlier, we employ browser instances to represent clients.

Fig. 4 and 5 display the outcomes of the second experiment, examining data size, training time, and conversion time. This experiment only completed one training iteration to produce a single model update. `No Comp.` denotes uncompressed data, whereas `No Enc.` denotes unencrypted model updates. We noticed that all compression setups produced convergent size outputs across all tested browsers, while there are discrepancies in conversion speed. Although encryptions increased data size by up to 136%, applying LZ-string compression reduced the data size to even smaller than the original raw data.

[2]https://developer.mozilla.org/en-US/docs/Web/API/Web_Crypto_API

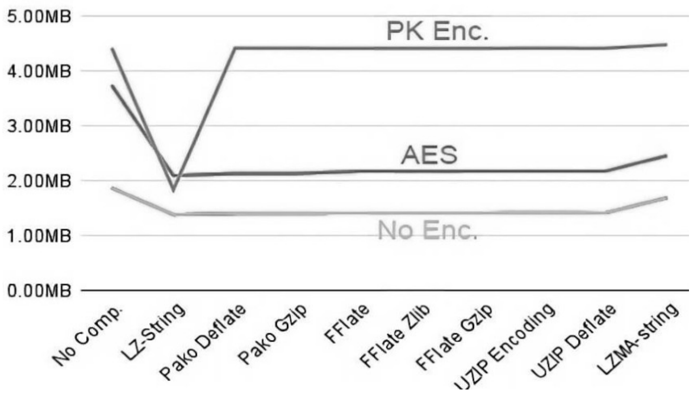


Fig. 4. Data size comparison (encryption against compression)

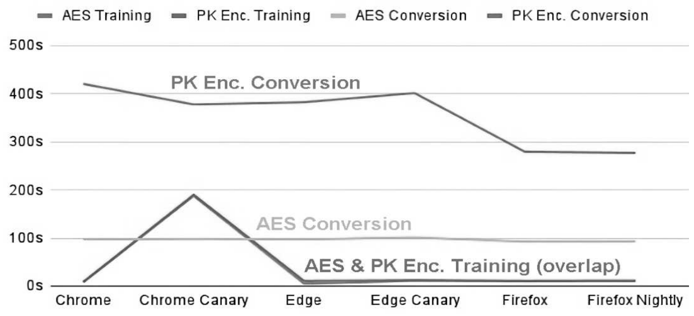


Fig. 5. Training and conversion time comparison (encryption against browser)

Remarkably, other compressions beside LZ-string did not perform well on PK cases. AES outputs a JSON object and PK produces data stream bytes. LZ-string generates a transmuted string, a serialised data that is converted to a higher Unicode, while the others typically output bytes, resulting in ineffeciency in reducing data size due to byte-to-byte conversion.

On Fig. 5, we performed 45 compression scenarios with PK but only 24 setups with AES due to data type restrictions. Hence, the conversion duration is not to be compared for both encryptions. The raw data was used during the training phase. Thus, encryptions and compressions did not affect the training duration, resulting in overlapping lines on AES and PK trainings. All browsers perform consistently on both training times, except Chrome Canary. Nonetheless, AES conversion time is converged in all instances. Notably, Firefoxs cut PK conversion time by up to 52% compared to other browsers.

C. Measurement of Memory Usage

From the previous experiments, we can see that achieving the same level of communication cost to the original FL without accuracy drawback is possible. However, employing additional techniques on top of the FL framework could compromise the performance. This set of experiments intends to evaluate the computation cost by measuring the memory usage. Taking performance into consideration, we employ the LZ-string compression and browser's performance measurer.

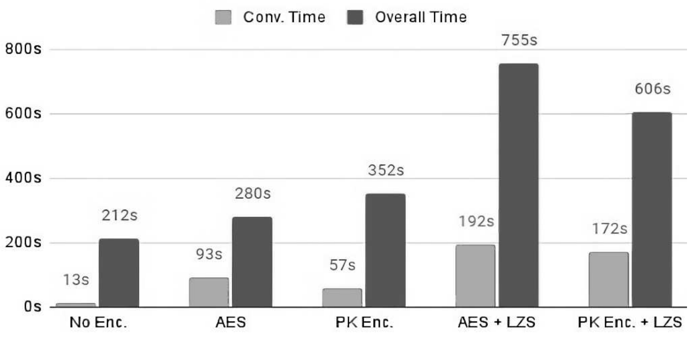


Fig. 6. Comparison of conversion time (encryption with compression)

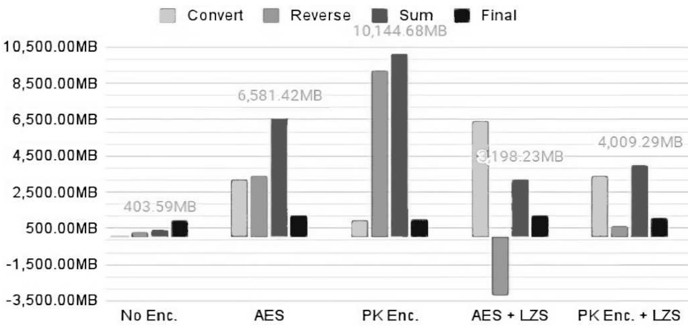


Fig. 7. Comparison of memory heap size (encryption with compression)

Fig. 6 and 7 depict the test results, which spanned the entire training process and involved numerous combinations of proxy, encryption, and compression. We assessed conversion time, training duration, and the cost of communication (incoming and outgoing) and computing (memory heap metrics).

We exclude communication time from our assessment because many operations operate asynchronously, enabling the machine to execute instructions concurrently with data transformation and transmission. The research aimed to focus on the client side, specifically measuring memory utilisation and bandwidth. We did not document processing time on the server and proxy sides to maintain anonymity, as it should not associate a specific request with a particular client.

While using compression can save half of the bandwidth, it prolongs the overall training time by threefold, mostly because its procedure takes longer than encryption. Due to local network operation, emitted data size does not impact overall time linearly. Further data translations from JSON objects to data stream bytes, and vice versa, occurred on AES with LZ-string, rendering it the slowest among the five setups.

Fig. 7 displays the memory consumption during the conversion phase (data transmutation, encryption, and compression), reversion phase (decryption and decompression), the sum of both phases and upon completion of the full round. Overall, AES consumes less memory than PK, while compression utilises more memory during the conversion phases than during the reversion phase. Employing encryption will increase memory utilisation, however, by applying compression we can reduce the memory heap usage by more than half.

Overall, symmetric AES consumes less memory than asymmetric PK, while compression utilises more memory in the conversion phases than in the reversion phase. It is expected that applying encryption will increase memory usage compared to the original one. Nevertheless, applying compression can help to cut the memory heap usage by more than half.

Remarkably, in the compression case, the AES encryption process led to a decrease in memory usage during the reverse operation, suggesting that the initial encryption and compression processes required substantial memory but then released some upon reversal. Suming up conversion and reversal heap sizes, the utilisation of compression on both encryptions used memory that was relatively similar, differing by 25.3%. At the conclusion of the training, memory utilisation across all five setups is comparable, with a standard deviation of 10%.

D. Real Network Test

All prior tests were conducted in a local network; thus, this experiment aims to assess the performance of the suggested technique in a real network environment. 48 tests, 12 clients, represented by browser instances, will interact with a real server via two arbitrary anonymising proxies on the internet.

TABLE III
PROXIED FL IN LOCAL AND INTERNET ENVIRONMENT

Mode	Acc	Time	Comm.	Conv.	Heap
AES Local	97.4	111s	27.8MB	22.4%	297.2MB
AES Online	96.9	147s	27.8MB	16.4%	286.7MB
PK Local	97.3	100s	24.8MB	15.0%	275.1MB
PK Online	97.4	126s	24.8MB	11.5%	293.9MB

A total of 48 tests were done, combining AES and PK encryptions with LZ-string compression in local and online environments. Table III shows that both local and online setups for the proposed approach yield roughly similar training accuracy, communication costs, and heap memory usage. The main difference is in the training time, with the online setups took about 26-32% more time than the locals. The internet's ratio of conversion time to training time is less than the local setting's. This suggests that the network environment, rather than the data translation process, affects the training's duration.

E. Scalability Simulation

This fifth set of tests will evaluate the scalability of the proxied privacy-enhancement, assessing its robustness with user counts ranging from four to 100. The assessments will be executed via non-GUI simulations to enhance inference speed.

Fig. 8 and 9 show the average values of 756 simulated clients, graded relatively (0-100%) to the highest value of each category. The training duration increases with the rise in users due to the shared resources in the simulation (not real devices with dedicated resources). This resource sharing also affects heap memory usage, which generally exhibits a slight downward trend. The test with 25 users has heap memory

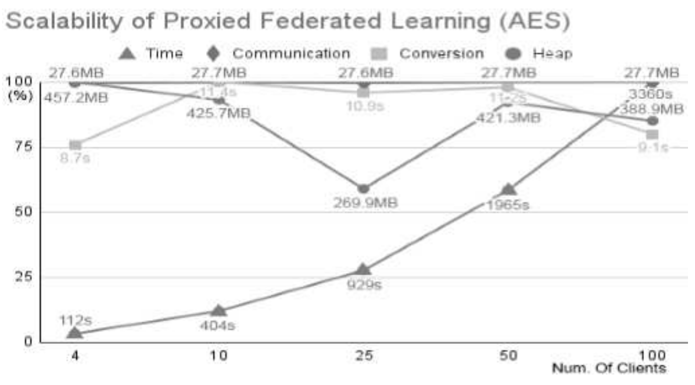


Fig. 8. Scalability of the proposed method in symmetric AES

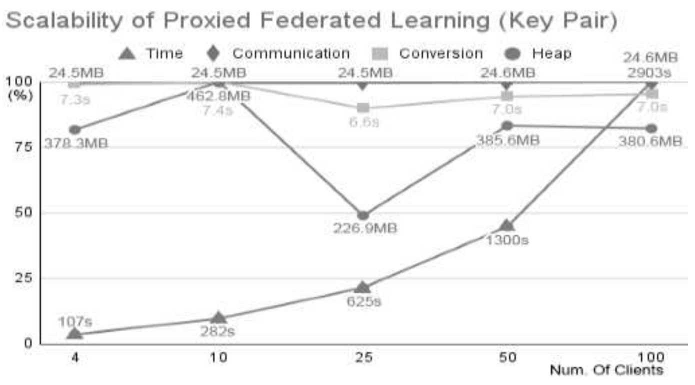


Fig. 9. Scalability of the proposed method in asymmetric PK

outliers due to undedicated resource. However, the overall comparison of heap memory and speed between AES and PK encryptions are relatively consistent, with PK generally trained faster and used lower memory compared to AES. The standard deviation for conversion time is 12.4% in the AES case and 4.2% in the PK case, whereas the communication cost is pretty similar throughout the tests for both cases. This shows that the technique yields consistent output irrespective of user scale.

F. Against Other Privacy-Preserving Techniques

The final segment of the experiments aims to compare the proposed approach with other prevalent privacy enhancements including differential privacy (DP), secure aggregation (SA), and homomorphic encryption (HE). 120 clients were employed, applying few configs such as clipping radius (CR), noise scale (NS), max share (MS), and polymod degree (PD).

Configs were picked to show the trend between them and the methods. The clipping radius aims to prevent exploding gradients, whilst the noise scale seeks to perturb data to maintain privacy. Both factors affect the accuracy. The maximum share influences secret sharing's value among peers. CKKS with TC128 scheme was used to support float numbers on the model updates, whereas polymod affects the ring structure for ciphering. Lastly, our proposed technique was assessed based on the prior encryptions and compression techniques.

Based on Table IV and Fig. 10, augmenting the clipping radius and noise scale impaired the accuracy down to 41%. In the case of secure aggregation, changes in the clipping radius

TABLE IV
COMPARISON OF PRIVACY-PRESERVING TECHNIQUES

Tech.	Config	Acc	Time	Comm.	Heap
DP	CR=0.1, NS=0.1	98.2	313s	236MB	860MB
DP	CR=0.5, NS=0.5	41.3	316s	237MB	883MB
SA	CR=0.1, MS=1	98	485s	1393MB	164MB
SA	CR=0.5, MS=5	97.8	450s	1391MB	200MB
HE	CKKS-TC128, 4096	97.5	860s	1949MB	5303MB
HE	CKKS-TC128, 32768	97.5	945s	2052MB	5623MB
PC	AES, no compress	97.6	218s	472MB	388MB
PC	PK, no compress	97.7	282s	558MB	307MB
PC	AES, LZ-String	97.9	782s	265MB	266MB
PC	PK, LZ-String	97.4	671s	232MB	251MB

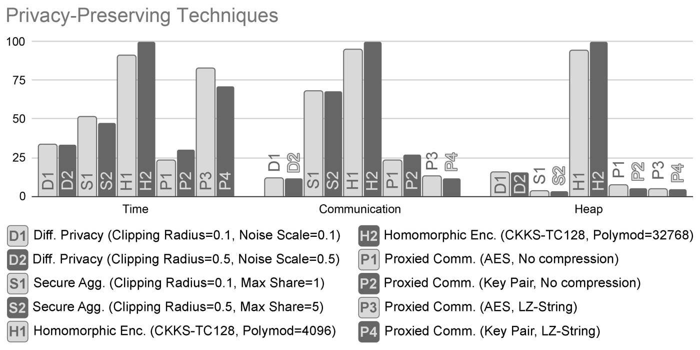


Fig. 10. Comparison of privacy-preserving techniques

and max share had minimal impact on accuracy; however, it increased the heap memory used by 12%. As for homomorphic encryption, the increase in polymod resulted linearly to training time, communication cost, and heap memory size. Applying compression on proxied communication extended training time but reduced communication cost and heap memory used.

Considering accuracy, only differential privacy has accuracy drawback. Given the training length, proxied communication without compression and differential privacy are among the fastest and can be deemed effective in performance. In terms of bandwidth, differential privacy and proxied communication with compression yielded the most efficient outputs, while other methods incurred costs up to eightfold higher. Lastly on heap usage, secure aggregation and proxied communication with compression are regarded as the most efficient in computation. Based on evaluations, we consider the proxied communication without compression is the most comprehensive among privacy-preserving methods, exhibiting superior performance speed and efficiency in communication and computation. Conversely, homomorphic encryption has the slowest performance and inefficient communication and computation.

V. EVALUATION

A. Mitigating Threats and Risks

1) Accuracy loss: The method does not employ perturbance or noise, thus, there is no accuracy drawback by default. Package lost is not seen either in our experiment. Due the encrypted content being sent, package lost will invalidate the payload. When it happens, the server will return an invalid response and the client can resend the data.

2) Server-proxy collusion: Collusion between server and proxy will undermine the efficacy of anonymisation, exposing the source's identity. Server puts bounty for revealed identity is not uncommon practice. To mitigate this, clients should refrain from utilising unreliable servers for inference and choose dependable proxies, with private proxies being preferable. Clients may also use a distinct proxy for each transmission.

3) Server-client collusion: In some cases, such as secure aggregation and multi-party computation, a server or aggregator may conspire with some clients to identify a certain client. This may occur in 2SC where the server selects participants and colludes with some counterfeits to determine a specific client's updates. To avoid this practice, clients should use the stronger 3SC technique and avoid employing unreliable server.

4) Client tracing: To trace a client, a server may send unique public key to each client and try to match the incoming data with specific decryptor. This adversary can on 2SC server's key pair and 3SC model training key pair (server's second key). Likewise, the server may also give different endpoint for each client. This method may jeopardise client privacy; however, the server may gain some benefits such as stronger authentication, participation monitoring, dropout prevention, heterogeneity identification, and imposing training rules. Nonetheless, this conduct may only be feasible with few clients due to heavy try-and-match computation. Clients are advised to ensure the key pair and target URL are public (not client-specific) and to not reuse previous client's keys. Employing a second or third layer of proxies (e.g. Client-Proxy 1-Proxy 2-Server), similar to Tor network, will enhance more anonymity to the source's identity, in the sacrifice of speed.

5) Content integrity: A threat may arise in the 2SC approach if the server sends the key to the client via a proxy rather than directly. By swapping with their own key, the proxy can read and alter the transmitted data from both sides. This may also occur in the 3SC method if the client employs the same proxy for both the model request and update transmissions. Clients are advised to utilise proxies in a rotating manner, using a distinct proxy for each communication, and opt for a private or reliable proxy. Using a simple encoding algorithm that allow the proxy to break should also be avoided.

6) Dropout and exploitation: Applying a voluntary system for capable clients in the 3SC method can mitigate dropout risks and participation abuse. In the case that the chosen proxy is non-responsive, the client may use an alternative proxy.

B. Optimised Configurations

Putting personal device in context, evaluation of the experiments mainly focuses on communication cost, computation overhead, and speed. While there is limitation for the proposed method to excel in all three criteria, there are some combinations to achieve specific needs:

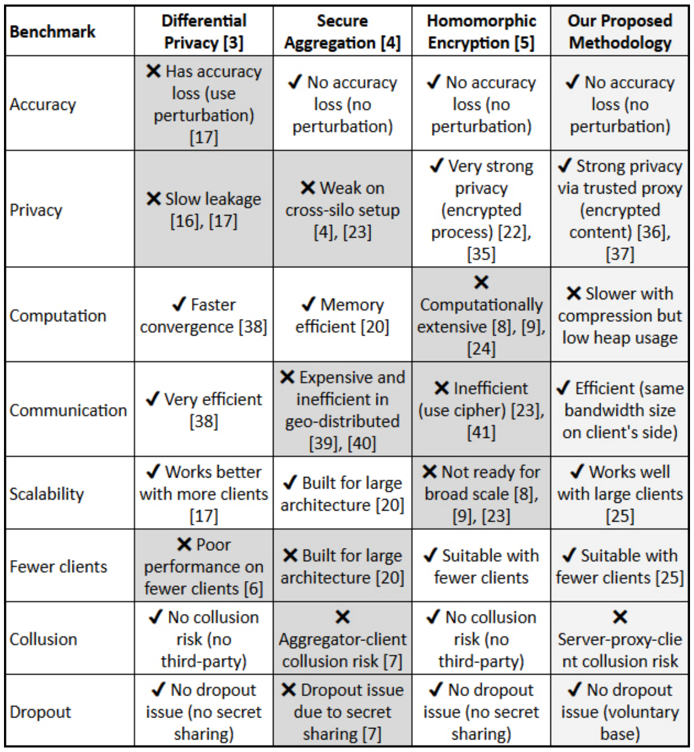
1) Higher security: Applying a robust encryption such as asymmetric encryption, avoiding key reuse, and employing rotating reliable or private proxies will enhance data security.

2) Higher performance and low communication cost: The use of effective compression methods, such as LZ-string compression, minimises computation and communication cost.

3) Faster inference and limited resource: Implementing compression prolongs training time, particularly when combined with robust encryption. A simpler encryption method without compression enhances inference speed and is suitable for devices with limited computational resources.

C. Privacy Enhancement Techniques Benchmark

We evaluate and benchmark the four privacy enhancements, including ours, based on aspects referring to Fig. 11:



Benchmark	Differential Privacy [3]	Secure Aggregation [4]	Homomorphic Encryption [5]	Our Proposed Methodology
Accuracy	✗ Has accuracy loss (use perturbation) [17]	✓ No accuracy loss (no perturbation)	✓ No accuracy loss (no perturbation)	✓ No accuracy loss (no perturbation)
Privacy	✗ Slow leakage [16], [17]	✗ Weak on cross-silo setup [4], [23]	✓ Very strong privacy (encrypted process) [22], [35]	✓ Strong privacy via trusted proxy (encrypted content) [36], [37]
Computation	✓ Faster convergence [38]	✓ Memory efficient [20]	✗ Computationally extensive [8], [9], [24]	✗ Slower with compression but low heap usage
Communication	✓ Very efficient [38]	✗ Expensive and inefficient in geo-distributed [39], [40]	✗ Inefficient (use cipher) [23], [41]	✓ Efficient (same bandwidth size on client's side)
Scalability	✓ Works better with more clients [17]	✓ Built for large architecture [20]	✗ Not ready for broad scale [8], [9], [23]	✓ Works well with large clients [25]
Fewer clients	✗ Poor performance on fewer clients [6]	✗ Built for large architecture [20]	✓ Suitable with fewer clients	✓ Suitable with fewer clients [25]
Collusion	✓ No collusion risk (no third-party)	✗ Aggregator-client collusion risk [7]	✓ No collusion risk (no third-party)	✗ Server-proxy-client collusion risk
Dropout	✓ No dropout issue (no secret sharing)	✗ Dropout issue due to secret sharing [7]	✓ No dropout issue (no secret sharing)	✓ No dropout issue (voluntary base)

Fig. 11. Benchmark between privacy enhancement techniques

1) Accuracy and Privacy: Differential privacy's accuracy is compromised by data perturbation [17], while there is no accuracy loss on other methods due to no noise added.

In regards to privacy, differential privacy faces challenges with gradual leaking and potential data leakage in a stationary user base [16], [17]. secure aggregation is ineffective in cross-silo and few-client configurations [4], [23], whilst homomorphic encryption is considered the pinnacle of cryptography, offering the utmost privacy protection [22], [35]. Our proposed methodology focuses on achieving strong privacy through the use of a trusted proxy alongside secure encryptions like XSalsa20 and Poly1305, as utilised in our studies [36], [37].

2) Computation and Communication Cost: Differential privacy is efficient in communication and features a faster convergence as it is independent from a third party [38], but used more memory heap. Conversely, secure aggregation is more memory efficient [20] but expensive in communication because it involves additional aggregator and frequent interactions [39]. It also faces difficulties on implementation in a geographically distributed setting [40]. Present homomorphic encryption remains computationally inefficient due to the extensive computing needed for the ciphering process [8], [9], [24]. This method is also deemed inefficient in terms of transmission cost when the size of the output data is up to more than five times (in our case, 20 times) larger than the initial data size [23], [41]. Although the methodology we propose involves another layer of communication, but unlike secure aggregation, it does not require additional interactivity in the side of client. This methodology can achieve the same degree of bandwidth size on the user's side as the original federated learning when compression is applied, as in our experiment. While the heap memory usage is comparable to the original, the technique is slow in inference and consumes up to 10 times more memory than the original throughout the process. Hence, we believe that this methodology requires enhancement, particularly in terms of computational efficiency.

3) Scalability and Working with Fewer Clients: Differential privacy is more suitable for larger user bases due to its greater generalisation and less customisation [17]. However, this method is ineffective with few clients due to noise injection, making it unreliable for providing accurate inferences with small data amount [6]. Secure aggregation is scalable but faces constraints when dealing with a small number of clients due to its main emphasis on large-scale architectures [20]. Current homomorphic encryption is not ready for broad adoption due to its significant computational load [8], [9] and communication costs [23], but it is fine with few clients. Our experiments show that our method run efficiently with both small and large user bases due to its simplified configuration, influenced by the operations of Oblivious DOH [25].

4) Collusion and Dropout: Collusion poses a significant risk for the proposed approach and Secure Aggregation. Other techniques are not vulnerable to this adversary as they do not involve a third party. Collusion may happen between aggregator and clients in secure aggregation case [7], and between server-proxy and server-client in our suggested method. Both cases aim to disclose certain clients' data. Similarly, if a party, such as a hacker, has access to both the proxy and the server, they can reveal a user's identity and content. Our mitigation varies from Secure Aggregation in that we can lessen the danger of collusion by selecting an arbitrary proxy.

If the server provides a "bounty" for disclosing source's identity, the proxy may be tempted to reveal source's identity in return for rewards. Therefore, it is recommended to use a trusted or private dedicated proxy to prevent collusion issues. Mitigation of the risks is detailed on the previous sub-chapter.

Unlike others, secure aggregation is affected by dropout and synchronisation issues [7]. Our approach utilises a voluntary system based on the 3SC algorithm, avoiding those issues.

D. Collaboration Between Approaches

Each privacy-preservation methods possesses its own advantages and limitations. Combining various methods can enhance privacy and security outcomes. The integration of differential privacy and proxy ensures privacy protection internally via perturbation and externally through anonymisation. Substituting the aggregator with a proxy in secure aggregation can streamline the process and provide a safer aggregation method. Homomorphic encryption features a robust algorithm that is mathematically proven and does not require an anonymisation process. However, proxies can facilitate the distribution of keys between servers and clients to enhance management efficiency.

VI. Conclusion

From our experiments and analysis, we can make inferences. Firstly, privacy-preserving techniques that maintain accuracy are an essential objective, particularly on personal devices with limited capabilities, followed by efficiency in computation and communication. Our proposed methodology has proven effective in meeting the specified criteria; however, further enhancements are necessary to expedite inference. While individual privacy enhancements have drawbacks and limits, combining many methods can synergistically preserve privacy while maintaining accuracy effectively and efficiently. This is particularly crucial in the personal device domain, where accuracy, privacy, and efficiency are all paramount.

Secondly, our tests demonstrate that it is feasible to achieve equivalent communication costs between our suggested encrypted anonymous privacy enhancement and the original federated learning through the utilisation of compressions on the model updates. Clients can enjoy enhanced privacy and data protection with anonymity and encryption without compromising their bandwidth. Additionally, using compression when applying encryption can reduce memory usage by up to 60% compared to setups without compression. However, the drawback is that the overall training duration increases when compression is implemented, causing a slowdown of up to 1.7 times. When implementing compression procedure, we suggest utilising Public-key asymmetric encryption instead of AES symmetric encryption for various advantages: 1) Reduce communication expenses and data volume; 2) Accelerate training iterations; and 3) Enhance security and safety. Compression is advised for personal devices to conserve bandwidth and memory when quick inference is not the main need.

Thirdly, every method has their risks and threats, for example, collusion between server-proxy-client, client tracing, and content alteration in our case. We have discussed how to mitigate these challenges in previous chapter. Our methodology currently cannot simultaneously achieve quick inference, bandwidth efficiency, and low memory usage. However, we have discussed previously options for specific needs. For example, applying compression can save memory and bandwidth, while without it will have good balance on speedy inference and fairly efficient in communication and computation.

Finally, the suggested methodology shows promising performance, surpassing popular techniques in certain parameters. Nonetheless, potential collaborations across methodologies exist to enhance robustness in privacy preservation and client data security, while achieving high accuracy, fast inference, and computation and communication efficient. We strongly believe that this research deserves further exploration and study due to its introduction of a novel approach for conducting anonymous federated learning via proxy on personal devices.

References

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep

networks from decentralised data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[2] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al . 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning* 14, 1–2 (2021), 1–210.

[3] Cynthia Dwork, Aaron Roth, et al . 2014. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci*. 9, 3-4 (2014), 211–407.

[4] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1175–1191.

[5] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 169–178.

[6] Robin C Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv*:1712.07557 (2017).

[7] Swanand Kadhe, Nived Rajaraman, O Ozan Koyluoglu, and Kannan Ramchandran. 2020. Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning. *arXiv preprint arXiv*:2009.11248 (2020).

[8] Cem Dilmegan. 2022. What is Homomorphic Encryption? Benefits & Challenges [2022]. (April 2022). https://research.aimultiple.com/homomorphic-encryption/ (Accessed on 24/02/2024).

[9] Wei Wang, Yin Hu, Lianmu Chen, Xinming Huang, and Berk Sunar. 2013. Exploring the feasibility of fully homomorphic encryption. *IEEE Trans. Comput.* 64, 3 (2013), 698–706.

[10] Jakub Konečn ‘y, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).

[11] Jakub Konečn ‘y, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527* (2016).

[12] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. 2019. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 13, 3 (2019), 1–207.

[13] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *Advances in Neural Information Processing Systems* 32 (2019). Manuscript submitted to ACM

[14] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 691–706.

[15] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 3–18.

[16] Johannes Gehrke, Edward Lui, and Rafael Pass. 2011. Towards privacy for social networks: A zero-knowledge based definition of privacy. In *Theory of cryptography conference*. Springer, 432–449.

[17] Morten Dahl. 2016. Differential Privacy for the Rest of Us — by Morten Dahl — Snips Blog — Medium. *Medium* (July 2016). https://medium.com/snips-ai/differential-privacy-for-the-rest-of-us-665e053cec17 (Accessed on 18/02/2024).

[18] Olivia Choudhury, Aris Gkoulalas-Divanis, Theodoros Salonidis, Issa Sylla, Yoonyoung Park, Grace Hsu, and Amar Das. 2019. Differential privacy-enabled federated learning for sensitive health data. *arXiv preprint arXiv:1910.02578* (2019).

[19] di Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.

[20] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečn ‘y, Stefano Mazzocchi, Brendan McMahan, et al . 2019. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems* 1 (2019), 374–388.

[21] Justin Patriquin. 2019. Federated Learning with Secure Aggregation in TensorFlow — by Justin Patriquin — Cape Privacy (Formerly Dropout Labs) — Medium. *Medium* (December 2019). https://medium.com/dropoutlabs/federated-learning-with-secure-aggregation-in-tensorflow-95f2f96ebecd (Accessed on 16/02/2024).

[22] David J Wu. 2015. Fully homomorphic encryption: Cryptography's holy grail. XRDS: Crossroads, *The ACM Magazine for Students* 21, 3 (2015), 24–29.

[23] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. 2020. BatchCrypt: Efficient homomorphic encryption for Cross-Silo federated learning. In *2020 USENIX annual technical conference (USENIX ATC 20)*. 493–506.

[24] Wei Wang, Yin Hu, Lianmu Chen, Xinming Huang, and Berk Sunar. 2013. Exploring the feasibility of fully homomorphic encryption. *IEEE Trans. Comput.* 64, 3 (2013), 698–706.

[25] Eric Kinnear, Patrick McManus, Tommy Pauly, Tanya Verma, and Christopher A. Wood. 2022. RFC 9230 - Oblivious DNS over HTTPS. *IETF* (April 2022). https://datatracker.ietf.org/doc/rfc9230/ (Accessed on 25/02/2024).

[26] Sudheesh Singanamalla, Tanya Verma. 2020. Improving DNS Privacy with Oblivious DoH in 1.1.1.1. *Cloudflare* (December 2020). https://blog.cloudflare.com/oblivious-dns/ (Accessed on 21/01/2025).

[27] Guillaume Rosinosky, Simon Da Silva, Sonia Ben Mokhtar, Daniel Négru, Laurent Réveillère, and Etienne Rivière. 2021. PProx: efficient privacy for recommendation-as-a-service. In *Proceedings of the 22nd International Middleware Conference*. 14–26.

[28] Shivam Kalra, Junfeng Wen, Jesse C Cresswell, Maksims Volkovs, and Hamid R Tizhoosh. 2021. ProxyFL: Decentralized Federated Learning through Proxy Model Sharing. *arXiv preprint arXiv:2111.11343* (2021).

[29] Hongrui Shi, Valentin Radu, and Po Yang. 2023. Distributed Training for Speech Recognition using Local Knowledge Aggregation and Knowledge Distillation in Heterogeneous Systems. In *Proceedings of the 3rd Workshop on Machine Learning and Systems*. 64–70.

[30] EPFL. 2022. epfml/disco: Decentralized & federated privacy-preserving ML training, using p2p networking, in JS. https://github.com/epfml/disco. (Accessed on 10/02/2024).

[31] Chris Deotte. 2021. 25 Million Images! [0.99757] MNIST Kaggle. https://www.kaggle.com/code/cdeotte/25-million-images-0-99757-mnist. (Accessed on 10/02/2024).

[32] Mozilla. 2023. Performance: measureUserAgentSpecificMemory() method - Web APIs — MDN. https://developer.mozilla.org/en-US/docs/Web/API/Performance/measureUserAgentSpecificMemory(2023). (Accessed on 06/02/2024).

[33] Maarten Bodewes. 2016. hash - What is the maximum size of the plaintext message for RSA OAEP? - Cryptography Stack Exchange. https://crypto.stackexchange.com/questions/42097/what-is-the-maximum-size-of-the-plaintext-message-for-rsa-oaep. (Accessed on 10/02/2024).

[34] Muhammad Senoyodha Brennaf, Po Yang, and Vitaveska Lanfranchi. 2023. Communication Efficient on Symmetric and Asymmetric Encrypted Anonymous Federated Learning. In *2023 IEEE International Conference on Cryptography, Informatics, and Cybersecurity (ICoCICs)*. IEEE, 133–138.

[35] Wencheng Yang, Song Wang, Hui Cui, Zhaohui Tang, and Yan Li. 2023. A Review of Homomorphic Encryption for Privacy-Preserving Biometrics. *Sensors* 23, 7 (2023), 3566.

[36] Nicky Mouha and Bart Preneel. 2013. Towards finding optimal differential characteristics for ARX: Application to Salsa20. *Cryptology ePrint Archive* (2013).

[37] Daniel J Bernstein. 2011. Extending the Salsa20 nonce. In *Workshop record of Symmetric Key Encryption Workshop*, Vol. 2011.

[38] Wei-Ning Chen, Ayfer Ozgur, Graham Cormode, and Akash Bharadwaj. 2023. The communication cost of security and privacy in federated frequency estimation. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 4247–4274.

[39] Wenliang Du, Yunghsiang S Han, and Shigang Chen. 2004. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proceedings of the 2004 SIAM international conference on data mining*. SIAM, 222–233.

[40] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 1–19.

[41] Saransh Gupta, Rosario Cammarota, and Tajana Šimunić Rosing. 2022. Memfhe: End-to-end computing with fully homomorphic encryption in memory. *ACM Transactions on Embedded Computing Systems* (2022).

[42] M. Brennaf, P. Yang, V. Lanfranchi, ”A Comparative Analysis of Federated Learning Techniques on On-Demand Platforms in Supporting Modern Web Browser Applications,” in *2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2023, pp. 2601–2606.