IEEE Open Journal of the Communications Society

Received XX Month, XXXX; revised XX Month, XXXX; accepted XX Month, XXXX; Date of publication XX Month, XXXX; date of current version 31 October, 2024.

Digital Object Identifier 10.1109/OJCOMS.2024.011100

Block Encryption LAyer (BELA): Zero-Trust Defense against Model Inversion Attacks for Federated Learning in 5G/6G systems

SUNDER A. KHOWAJA[1] *(Senior Member, IEEE)*, PARUS KHUWAJA [2], KAPAL DEV[3] *(Senior Member, IEEE)*, KESHAV SINGH[4] *(Senior Member, IEEE)*, XINGWANG LI[5] *(Senior Member, IEEE)*, NIKOLAOS BARTZOUDIS[6], AND CIPRIAN R. COMSA[7] *(Senior Member, IEEE)*

[1]Faculty of Computing, Digital and Data, Technological University Dublin, Ireland
[2]University of Sindh, Jamshoro, Pakistan
[3]Department of Computer Science, Munster Technological University, Ireland
[4]Institute of Communications Engineering (ICE), National Sun Yat-sen University, Taiwan
[5]School of Physics and Electronic Information Engineering, Henan Polytechnic University, China
[6]Centre Tecnològic Telecomunicacions Catalunya (CTTC/CERCA), 08860 Castelldefels, Spain
[7]Gheorghe Asachi Technical University of Iasi, Romania

CORRESPONDING AUTHOR: Kapal Dev (e-mail: kapal.dev@ieee.org).

This work was supported by Boosting Ingenium for Excellence (BI4E) project, funded by the European Union's HORIZON-WIDERA-2021-ACCESS-05-01-European Excellence initiative under the Grant Agreement No. 10071321.

ABSTRACT Federated Learning (FL) paradigm has been very popular in the implementation of 5G and beyond communication systems as it provides necessary security for the users in terms of data. However, the FL paradigm is still vulnerable to model inversion attacks, which allow malicious attackers to reconstruct data by using the trained model gradients. Such attacks can be carried out using generative adversarial networks (GANs), generative models, or by backtracking the model gradients. A zero-trust mechanism involves securing access and interactions with model gradients under the principle of ”never trust, always verify.” This proactive approach ensures that sensitive information, such as model gradients, is kept private, making it difficult for adversaries to infer the private details of the users. This paper proposes a zero-trust based Block Encryption LAyer (BELA) module that provides defense against the model inversion attacks in FL settings. The BELA module mimics the Batch normalization (BN) layer in the deep neural network architecture that considers the random sequence. The sequence and the parameters are private to each client, which helps in providing defense against the model inversion attacks. We also provide extensive theoretical analysis to show that the proposed module is integratable in a variety of deep neural network architectures. Our experimental analysis on four publicly available datasets and various network architectures show that the BELA module can increase the mean square error (MSE) up to 194% when a reconstruction attempt is performed by an adversary using existing state-of-the-art methods.

INDEX TERMS Zero-Trust, Model Inversion Attacks, Block Encryption LAyer, Federated Learning, 5G/6G systems.

I. INTRODUCTION

THE introduction of fifth generation (5G) communication systems brought the emerging technologies such as Internet of Things (IoT), artificial intelligence (AI), network slicing, software defined networks, network function virtualization, and smart sensors, together for advancing connectivity and fulfilling the user demands. For instance, IoT and smart sensors in 5G enable heterogeneous devices to connect to the network, acquire data in real-time, and use services related to connectivity, automation, and communication. Similarly, network slicing divides the network virtually for specific users and manages communication

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

between them. However, with the recent change of landscape concerning AI methods and the users inclined to use them in an ubiquitous manner, the demand for number of connections, latency requirements, and high bandwidth has been increased drastically [1], [2]. Many researchers have suggested that the 5G communication system will not be able to cater such demands while adapting to rapid developments in the field of AI [1], [2]. Researchers are working towards the development of beyond 5G networks to meet the requirements of large-scale networks, accommodate vast number of edge devices, and integrate seamlessly with AI technologies. This will significantly improve the efficiency and operation of communication networks.

Existing works have extensively proposed sixth generation (6G) communication systems to be the answer for the growing needs associated with 5G systems concerned with the bandwidth, connectivity, latency, and scalability [3]. However, researchers have also identified challenges that revolve around heterogeneity, softwarization of radio access networks, compliance with diverse technologies, complexity of aerial base stations, full integration and adoption of AI technologies, and security [4]. The security is one of the major problems assumed to be associated with 6G networks due to its interconnected and distributed nature. However, this interconnected nature of AI in 6G networks exposes the ecosystem to privacy breaches and cyberthreats, which could compromise sensitive data or infrastructure [3], [4]. For instance, consider an example of a smart healthcare system in a 6G-powered city that uses AI to support real-time healthcare services. Such services utilize wearable health monitors that continuously track vital signs and report to cloud-based AI systems for analysis, autonomous ambulances that navigate using AI powered by real-time traffic and environmental data, and AI-assisted surgery performed remotely through AR devices and tactile robotics. These systems are vulnerable to data poisoning attacks, where malicious attackers can inject poisoned data through wearable devices. An example of this would be false health alerts, such as fake cardiac arrhythmias sent to cloud servers, causing unnecessary medical responses. They are also vulnerable to model inversion attacks, which can exploit an AI model used for real-time navigation to reconstruct sensitive data, such as the location of hospitals or routes frequently used by ambulances.

Many studies have proposed the use of a distributed learning paradigm called Federated Learning to address data security issues as it allows the edge devices to train the model locally or collaboratively without sending the data to the server. The server then collects the gradients from the models that are locally computed in order to aggregate for constructing a global model, which is then shared back with the edge devices. In the above training cycle, the data never leaves the owner, which makes the Federated Learning paradigm interesting to the service provider as well as users. Although Federated Learning provides necessary privacy to the data, it is still vulnerable to the model privacy attacks such as model inversion that led to the reconstruction of sensitive data. A key previous work [5] introduced for the first time the attack on the gradients coining the term *Deep Leakage from Gradients* (DLG), whose goal is to reconstruct the data. Since the DLG proposal, many studies have tried to improve the attack efficiency [6]–[9] as well as propose a defense mechanism against the model inversion [10]–[12].

In addition, other works have observed the effects of model inversion attacks in the context of potential 5G / 6G communication networks [4] and vehicular networks [9]. The studies show that such an attack not only disrupts the communication services and harms an individual's identity by recreating sensitive information but also causes financial and emotional losses. The true gradient using the DLG was obtained by optimizing randomly initialized data with its associated label in an iterative manner. Once the training was complete, it was assumed that the randomly initialized data converged to the true gradient. Some existing studies [7] have pointed out DLG's limitation towards the sensitivity and training instability when varying resolution and batch size. An improved version of DLG (iDLG) [7] used the Softmax layer to analytically compute the ground-truth label using the gradient of the loss. A stable version of DLG named inverting gradient [6] was proposed that used a loss function based on magnitude-invariant cosine-similarity. Recently, an approach named generative model inversion (GGL) was proposed in [13] that aims to generate similar data while leveraging a pretrained GAN. Similar data are generated by assuming the true label (identical to iDLG) and then optimizing the generated output through matching gradients. It should be noted that GGL does not aim to reconstruct the original data, rather it generates a similar data that belongs to the data distribution, used for training the pretrained GAN. Such methods recover sensitive training data (e.g., images and labels) from gradients shared during collaborative training, such as FL. By leveraging a more accurate analysis of gradient updates, the aforementioned methods expose vulnerabilities in gradient-sharing protocols, highlighting the potential privacy risks in AI model training.

Similarly to the design of model inversion attacks, some studies have also proposed related defense mechanisms to cope with the aforementioned attacks in the context of 5G/6G communication and vehicular networks. These methods include data sanitization [14], [15], gradient perturbation [5], [10], intentional adversarial attacks [12], hybrid methods [17], [18], and fusion rating scheme [16], respectively. Some of the methods are either not effective to the model inversion attack or degrade the performance of the system enough to limit its realization in real world systems, especially when using cryptography. In this regard, we refer to a recent study [18] that highlighted the relationships between FL settings, its architectural design, and factors such as number of iterations, activation function, resolution, and batch size before the gradient exchange. The findings in the study are

This article has been accepted for publication in IEEE Open Journal of the Communications Society. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2025.3526768

compliant with some of the existing works as well, such as the assumption of the activation function to be twice differentiable and impact of modifying number of iterations on inversion performance. This study provides an insight regarding the transformations when applying encryption on a model in FL settings.

With the emergence of data and model security threats, researchers have emphasized the design of zero-trust security frameworks that are based on the principle of compulsive verification. From the data perspective, zero-trust frameworks continuously authenticate the data transmission along with periodic analysis to ensure the security. However, in terms of model security, a very limited literature is available that employ zero-trust security frameworks. A particular study that focuses on the verification process was proposed in [19]. The study proposed a concept of digital passports that is responsible for performing transformational parameters in a normalization layer through passport functions. A digital passport for deep learning network parameters is a cryptographic mechanism that embeds unique, verifiable credentials within a model's parameters to ensure authenticity, traceability, and protection against unauthorized modifications or theft. The method [19] is designed to verify the ownership of the neural network. The study highlights that if the ownership of the neural network is forged (fake digital passport function), severe degradation in network's inference performance will be observed.

Considering the model inversion problems in FL settings, we proposed a zero-trust block encryption layer network (BELANet) to provide defense against model inversion attacks. The study provides a theoretical basis that the feature maps obtained from batch normalization, convolutional, and fully-connected layers possess the information concerning reconstruction of data. The same information is also available with the gradients, especially when the backward propagation is carried out. We also show that the model inversion is only applicable when the gradient spaces between local and global models are aligned. Based on this assumption, we propose a zero-trust transformational block encoder that misaligns and secures the gradient spaces with an authentication mechanism while maintaining the aggregation process in FL settings. A randomly generated sequence for authentication mechanism and maintaining zero-trust is passed through fully connected layers, while mimicking as batch normalization layer. We show with our experiments that the proposed BELANet driven by the zero-trust principle is effective in defending the network against model inversion attacks. We also show that the reconstruction of data by the malicious attacker would not be possible without prior knowledge of the authentication mechanism and information concerning the BELA module parameters. The contributions of this work are summarized as follows:

- A zero-trust based BELANet is proposed for defense against model inversion attacks.

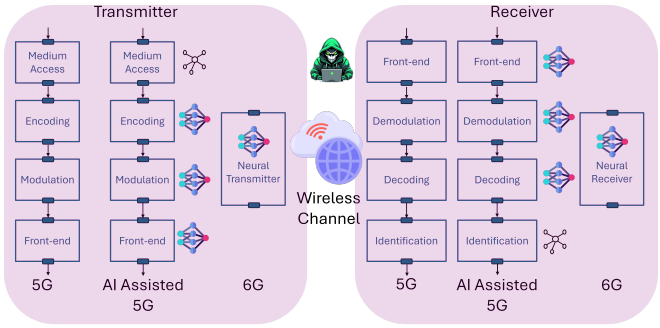

FIGURE 1. **Differentiating classical, AI assisted 5G, and 6G networks**

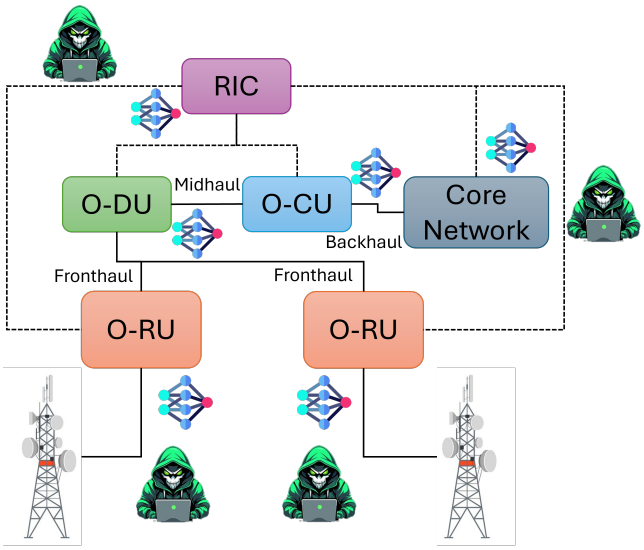

FIGURE 2. **AI-Assisted O-RAN architecture and associated vulnerabilities concerning AI-model attacks. O-RU: Open RAN Radio Units, O-DU: Open RAN Distributed Units, O-CU: Open RAN Centralized Unit, and RIC: RAN Intelligent Controller.**

- We proposed the BELA module that utilizes trainable encrypted parameters customized to each user in an FL setting.
- We provide theoretical analysis to show the validity of the BELA module's integration in deep neural network architectures.
- Extensive experiments are carried out to validate the efficacy of BELANet against model inversion attacks.

Subsequently, we structure the paper as follows: We consolidate the literature review based on the model inversion and defense methods in Section II. The details regarding the proposed methods are presented in Section III. Section IV provides an in-depth experimental evaluation and comparative analysis. Finally, the concluding remarks and potential future works are outlined in Section V.

II. Attack Vulnerabilities in 5G/6G

Although the model inversion attack is a general security vulnerability when it comes to the adoption of AI/ML techniques, the impact of such an attack in the overall system

depends on the network architecture and the application itself. As BELANet is proposed for 5G/6G systems, we establish a foundation to provide an abstract integration example in this section.

Although the native 5G networks did not use AI's integration in communication networks, the current 3rd Generation Partnership Project (3GPP) standardization, which maintains cellular standards, has proposed the integration of AI in 5G networks [36], [37]. The primary AI integration in 5G networks is mainly considered for air interface that could be applied in the domain of mobility optimization, load balancing, network energy saving, vehicular communication in autonomous vehicles, positioning, beam management, and channel-state information feedback. There are also other associated applications suggested in 3GPP Release 18; however, we have only listed a few of them. It should be noted that the 3GPP does not provide guidelines regarding AI model training or its development. Instead, it provides guidelines for building common evaluation methods and frameworks while integrating AI with air interface functions. 6G networks are still in their infancy, however, it is apparent that the AI/ML will be fundamental aspect of 6G communication systems. 6G is also considered to be an AI-native network, which suggests that the network is natively built in a way to run AI/ML models for all its related tasks. An example of the network design flow is shown in Figure 1, where the native 5G networks, AI assisted 5G networks and 6G networks are differentiated, accordingly. It can be visualized that the current 5G networks leverages AI/ML for each of the specific functions in order to optimize the performs. For the 6G networks, it is assumed that the entire air interface would use deep neural networks for achieving optimal performance.

Given the dependency of current 5G and 6G networks on AI services, it is apparent that models will be trained and transmitted to the receiver via the wireless channel. This is the basic assumption of a model inversion attack, where an attacker can potentially acquire the transmitted AI model and reconstruct the data. The BELANet can be used to make the trainable parameters private for each encoding, modulation, and front-end unit, preventing the attacker from reconstructing the data from the acquired trained model.

A similar case can be discussed for the Open Radio Access Network (O-RAN) alliance outside of 3GPP, which explores the use of AI/ML services for network management and orchestration. For instance, the O-RAN alliance has a RAN Intelligent Controller (RIC) that leverages AI/ML models to optimize applications. The RIC also hosts a variety of rApps and xApps, designed to run AI/ML services in near real-time and non-real-time. It is expected that the functionality of xApps and rApps will grow further with the expansion of the O-RAN ecosystem. In such a use case, where the RIC supports a number plate recognition app or an application that performs facial analysis, it is potentially vulnerable to model inversion attacks. If an attacker acquires just the trained model, it may lead to the privacy leakage of user data. An example of the AI-assisted O-RAN architecture is shown in Figure 2.

III. Related Works

This section provides a brief review of the literature on model inversion attacks and associated defenses in FL settings. The emergence of model inversion attacks that can reconstruct the private training data from compromised gradients pose a significant threat to the privacy preservation of collaborative and centralized training algorithms. Membership inference attack is associated with exploiting training data in a centralized learning setting [20] that determines if the data belongs to the training distribution or not. In collaborative learning systems, the seminal work DLG [5] was the first to propose an attack method for the reconstruction of data from compromised gradients. The study in [6] improved the DLG method in terms of stability and efficiency by proposed a loss function based on magnitude invariant cosine similarity. Khowaja et al. [9] improved the DLG method by using a better generative adversarial network (GAN) architecture to reconstruct the private data from model inversion. The study in [21] proposed the method of gradient inversion in alternative spaces (GIAS), which indicates that compromised gradients are not enough to recreate the data; therefore, they used a pre-trained model for revealing the data. The study in [8] proposed a way to obtain the ground-truth labels from the last fully connected layers of the network, which is then used to augment the compromised gradients using batch normalization statistics, thus improving the quality of data reconstruction. Most of the aforementioned methods were used for general recognition methods.

Some studies have put forth more efforts on the design of GAN methods to improve the reconstruction method from model inversion attacks. The study [22] proposed a GAN network that generates the dataset from model inversion which closely approximates the distribution of the training data. The study in [23] proposed mGAN-AI, a method that performs auxiliary identification using the proposed multitask discriminator and then generates the data from model inversion. The study in [24] designs a combined loss function along with encoders to optimize the shadow models and their gradients for main classification tasks. In this process, the shadow models are capable of generating features that could be used for recreating the private data. The study [9] improved the GAN architecture to design an attack mechanism to recreate the data and then used model poisoning to corrupt the samples. The model is then sent for the aggregation process in the FL setting, resulting in a degradation of performance with each passing communication round. GGL method [13] was also recently proposed that generates fake data using GANs whose parameter weights are kept fixed. The trainable parameters are then passed through GGL as input. The GGL

This article has been accepted for publication in IEEE Open Journal of the Communications Society. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2025.3526768

method then employs the Covariance Matrix Adaptation Evolution Strategy [13] to reduce the similarity between the private data and the GAN generated data. Some of the aforementioned methods have been proposed in the context of 5G/6G and vehicular communication networks, however, they are limited to the design of attack methods only.

In recent years, many studies have tried to develop an efficient method to propose defense against inversion attacks to improve model security. These methods include secure multiparty computation, homomorphic encryption, differential privacy, data sanitization, and gradient perturbation [4]. The study in [7] proposed the addition of Laplacian and Gaussian noise to prevent the model inversion. The experiments revealed that the model inversion attack failed when the noise variance exceeds 10^{-2}, however it also affects the model performance. In [25] a data perturbation method was proposed to protect the model from model inversion attacks that avoid compromising the model's performance. Similarly to the aforementioned approach, a differential privacy-based method was proposed in [17], which adds decay noise to the training data of each client. Although the noise injection helps in preventing the model inversion, it significantly degrades the model's inference performance as well. In addition, a computational cost overhead is added as the noise is injected into all training data from the client. Wang et al. [26] proposed a method Outpost that also depends on the addition of noise to the gradients in order to cope with model inversion attacks. The method uses Fisher Information Matrix to determine the level of noise to be added to the gradients at each iteration. The method suggests that it achieves a better trade-off in terms of resiliency to attacks and computational overhead. The method in [12] added an intentional adversarial attack at the client side to train the model. Subsequently, the method uses the proximal gradient layer to recover from the adversarial attack on the server side to improve the inference performance. Meanwhile, the attacker was unable to recreate the training data. However, the computational cost of training the PGSL is high in comparison to existing methods. In addition, PGSL is limited in terms of scalability, making it unsuitable for 5G/6G systems. The method in [16] proposed a zero-trust based self-supervised learning paradigm and generative AI methods to detect whether the sample belonged to training data distribution or not. During the downstream task, the method was trained with shared weights to detect poisoned or clean samples. The method provides an adequate defense against model poisoning, but is not adequate for model inversion attacks. Recently, a study [20] proposed the zero-trust method to defend against model inversion by training shadow models with joint loss function with the main classification model. The shadow models can either be used to attack the trained models or can be used to provide defense against model inversion. However, the computational cost for training such models is quite high, which makes it unrealizable for time-sensitive applications.

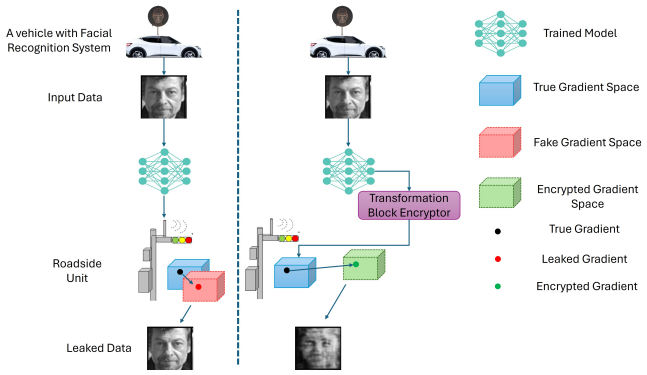

FIGURE 3. Depicting the impact of BELANet on model inversion. $\{Left\}$ **A smart car enabled with facial recognition feed the data to the training model and sends it to the roadside unit, where the malicious attacker reconstructs the data from the shared gradient.** $\{Right\}$ **A smart car enabled with facial recognition feed the data to the training model with Block Encryption Layer. The attacker is unable to reconstruct the data from the gradients.**

We also provide a comparative analysis between the proposed work and the existing defense methods in Table 1. Table 1 differentiates the proposed work from existing ones in terms of the method used, flexibility (the ability of the defense method to be applicable to different network architectures and domains), setting (the ability of the defense method to handle independent and identically distributed (IID) data and non-IID data), theoretical guarantee, and whether the parameters are trainable. It can be seen that the proposed method, along with Soteria and ZETA, are the only ones that propose the parameters to be trainable. To the best of our knowledge, this is the first work to provide a block encryption layer module based on zero-trust principles for defense against model inversion attacks using trainable encryption parameters customized for each user in FL settings, which is feasible for deployment in 5G/6G communication networks.

IV. Proposed Method

An overview of the BELANet module is shown in Figure 3. In this section, we provide the details regarding the correlation of gradient information with the input and associated labels during the backpropagation process. We further show that the reconstruction of data from model inversion can only be performed if the latent spaces are aligned. Using the above assumptions, we provide details regarding the proposed transformational block encryptor module to cope with model inversion attack. Lastly, we present details regarding the integration of the proposed module in FL settings.

A. GRADIENT CORRELATION

The model inversion attack assumes that the data can be reconstructed using gradients. Several studies including [6] have established the fact that the information regarding the

TABLE 1. A comparison of Existing defense approaches with the proposed one. The ✓,×,○ refers to the available, not available, and partially available respectively.

Approach	Defense Method	Flexibility	Setting	Theoretical Guarantee	Trainable
IDLG [7]	Input Perturbation	✓	×	✓	×
Chamikara et al. [25]	Input Perturbation	×	×	○	×
Wei et al. [17]	Input Perturbation	×	×	○	×
Outpost [26]	Input Perturbation Fisher Information Matrix	✓	✓	○	✓
Soteria [10]	Input Perturbation	○	○	✓	✓
PGSL [12]	Adversarial Attack	×	○	○	×
ZETA [24]	Shadow Models	×	○	○	✓
Sun et al. [38]	Parameter Encryption	✓	○	○	×
Kim and Park [39]	Masking, Clipping, Pruning and Noise	×	○	○	×
Proposed	BN Parameter Encryption, Zero-Trust Principles	✓	✓	✓	✓

input data is embedded in the derivative of the loss value obtained from the output of the fully connected (FC) layer. Similarly, the chain rule can be used on the gradient of other layers to independently compute the input of the FC layer. Let us consider that the input and its corresponding label are denoted by X, y, respectively. Considering X, y, we can compute the derivative of the function $\mathbf{F}$, also known as the gradient, using the Jacobian matrix as follows.

$$\frac{\partial \mathbf{F}}{\partial X} = \begin{bmatrix} \frac{\partial \mathbf{F_1}}{\partial X_1} & \cdots & \frac{\partial \mathbf{F_1}}{\partial X_p} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{F_q}}{\partial X_1} & \cdots & \frac{\partial \mathbf{F_q}}{\partial X_p} \end{bmatrix} \tag{1}$$

where p and q represent the length of the vectors. The chain rule is then applied to the gradients, i.e., $\mathbf{G}(\mathbf{F}(X))$ suggesting that the inner function is differentiable with respect to X and the outer function is continuous and differentiable, respectively. The chain rule can be mathematically expressed as $\frac{\partial \mathbf{G}}{\partial X} = \frac{\partial \mathbf{G}}{\partial \mathbf{F}} \frac{\partial \mathbf{F}}{\partial X}$. In order to show the correlation between the gradients, input and associated labels, let us assume a linear regression example having a mean square error loss function that is represented as $Loss(X, y) = \frac{\sum(\vartheta \cdot X - y)^2}{Q}$, where ϑ represents weights and Q refers to the number of samples. The derivative of $Loss(X, y)$ with respect to ϑ can then be computed as follows.

$$\begin{aligned} \frac{\partial Loss}{\partial \vartheta} &= \frac{\partial}{\partial \vartheta} \frac{\sum_q (\vartheta \cdot X_q - y_q)^2}{Q} \\ &= \frac{1}{Q} \sum_q 2 \cdot (\vartheta \cdot X_q - y_q) \cdot \frac{\partial (\vartheta \cdot X_q - y_q)}{\partial \vartheta} \\ &= \frac{\sum_q 2 \cdot X_q \cdot (\hat{y} - y_q)}{Q} \end{aligned} \tag{2}$$

The above derivation provides the basis that the input data, its associated label, and the gradients are positively correlated. Thus, we can conclude that the model inversion can be used to reconstruct the input data.

The above finding remains valid for a classification task with a non-linear neural network, as we use the same methods for gradient computation to reduce the error. Let us consider a neural network with an activation layer, two FC layers, and one softmax layer. The model considering the forward propagation and cross entropy loss function can be mathematically described as $Loss = CE(\hat{y}, y)$. The notation $\hat{y}$ describes the predicted output which can be written as $\hat{y} = softmax(z)$. Here z is the function of input feature map $fmap$, parameter weights $\rho \in \mathbb{R}^{Q_c \times Q_{hn}}$, and bias $b_z \in \mathbb{R}^{Q_c \times 1}$, which can be expressed as $z = fmap \cdot \rho + b_z$. The feature map is derived from the rectified linear unit (ReLU) activation function, i.e. $fmap = ReLU(v)$, where v is the function of input data X, parameter weights $\vartheta \in \mathbb{R}^{Q_{hn} \times Q_X}$, and bias $b_v \in \mathbb{R}^{Q_{hn} \times 1}$. The notations Q_c and Q_{hn} correspond to the number of classes and number of hidden nodes, respectively. The derivative of the ReLU function $\frac{\partial ReLU(X)}{\partial X} = sgn(ReLU(X))$, which suggests that it would yield 1 if $X > 0$ and 0, otherwise. Considering that the derivative of loss function yields $(\hat{y} - y)^T$, the output layer gradient using chain rule can be mathematically written as

$$\frac{\partial Loss}{\partial b_z} = (\hat{y} - y)^T \tag{3}$$

$$\frac{\partial Loss}{\partial \rho} = fmap \cdot (\hat{y} - y)^T \tag{4}$$

The above derivation of the gradients are compliant with the observations in studies [7], [8]. Equation 3 shows that the gradients provide the information regarding labels while Equation 4 shows that the gradients embed the information regarding the feature maps, which can be used to recreate the input in FL settings as follows

$$fmap = \begin{cases} X \cdot \vartheta + b_v, & \text{if } X \cdot \vartheta + b_v > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$\begin{aligned}\frac{\partial Loss}{\partial \vartheta} &= \frac{\partial Loss}{\partial z}\frac{\partial z}{\partial fmap}\frac{\partial fmap}{\partial v}\frac{\partial v}{\partial \vartheta} \\ &= (\hat{y}-y)^T \cdot \rho \cdot sgn(fmap) \cdot X \\ &= \begin{cases}(\hat{y}-y)^T \cdot \rho \cdot X, & \text{if } fmap > 0 \\ 0, & \text{otherwise}\end{cases}\end{aligned} \tag{6}$$

$$\begin{aligned}\frac{\partial Loss}{\partial b_v} &= \frac{\partial Loss}{\partial z}\frac{\partial z}{\partial fmap}\frac{\partial fmap}{\partial v}\frac{\partial v}{\partial b_v} \\ &= (\hat{y}-y)^T \cdot \rho \cdot sgn(fmap) \\ &= \begin{cases}(\hat{y}-y)^T \cdot \rho, & \text{if } fmap > 0 \\ 0, & \text{otherwise}\end{cases}\end{aligned} \tag{7}$$

where $\rho = \rho' - \zeta(\hat{y}-y)^T \cdot fmap$ represents the current iteration weight while the ρ' refers to the previous iterations' weight, and ζ represents the learning rate. The above derivation is based on the assumption that b_v and ϑ are known in the FL system, which is typically the case. Considering the above derivation, it can simply be deduced that there is a positive correlation between the gradients, input, and associated labels, and also the model inversion from each layer can help in reconstructing X.

A convolutional neural network (CNN) architecture consists of two changes compared to the one described above. The first is the cross-multiplication, i.e., $v = \vartheta \otimes X + b_v$, and the addition of a flatten layer after the activation function, i.e., $fmap' = flatten(fmap)$. It can be deduced from the above derivations that the gradients will still be positively correlated with the data and associated labels. Therefore, it can be deduced that the data can be reconstructed from the model inversion attacks. The assumptions concerning the correlation of gradients with the data and labels are in accordance with some of the existing works [7], [8], which also suggests that the quality of reconstruction increases with the inference accuracy.

Recently, many studies suggest to employ a batch normalization layer in convolutional neural networks to address the problem of vanishing gradients which occurs when the values from the input are saturated concerning the activation function [7], [8]. Values are normalized with normal distribution using the batch normalization layer, which helps to cope with the vanishing gradients problem. We want to show that the gradients are positively correlated with the data and associated labels, even when the batch normalization layer is employed in the CNN. Let us represent $BN = \{v_1 \cdots v_M\}$ a minibatch, where M represents the size of the minibatch. The BN layer is defined by the mean $\mu_{BN} = \frac{1}{M}\sum_m v_m$ and variance $\sigma^2_{BN} = \frac{1}{M}\sum_m (v_m - \mu_{BN})^2$. We can define the output of the BN function as $BN = \beta + \hat{v} \cdot \gamma$, where β and γ are the hyperparameters associated with the BN layer. The notation $\hat{v}$ is defined as $\frac{v-\mu_{BN}}{\sqrt{\sigma^2_{BN}+\epsilon}}$, where ϵ is added to avoid getting ∞ as result. Based on the BN layer computation, we can use the same network that we defined for the deep neural network and adding further the BN layer and the flatten layer for representing a convolutional neural network with batch normalization layer. The BN layer will be added after the first convolutional layer, i.e. $BN = \mathbb{F}_{\beta,\gamma}(v)$, and the flatten layer will be added after the $fmap$ layer, which is represented as $fmap' = flatten(fmap)$. The minor changes to the deep neural network can be made as pointed out above for the CNN description. We can then compute the derivative of the loss function with respect to the hyperparameters of the BN layer as follows

$$\frac{\partial Loss}{\partial \beta} = \frac{\partial Loss}{\partial BN}\frac{\partial BN}{\partial \beta} = \rho \cdot sign(fmap) \cdot (\hat{y}-y) \tag{8}$$

$$\frac{\partial Loss}{\partial \gamma} = \frac{\partial Loss}{\partial BN}\frac{\partial BN}{\partial \gamma} = \rho \cdot sign(fmap) \cdot (\hat{y}-y) \cdot \hat{v} \tag{9}$$

The aforementioned equations indicate that the gradients from the BN layer are positively correlated with the input and associated labels. However, we need to verify if the gradient of the network is positively correlated with the data and associated labels. To do so, we must consider the input of the BN layer to compute the derivative of the loss function, i.e. $\frac{\partial Loss}{\partial v}$. In order to apply the chain rule for computing $\frac{\partial Loss}{\partial v}$, we need to determine $\frac{\partial Loss}{\partial \mu_{BN}}$, $\frac{\partial Loss}{\partial \sigma^2_{BN}}$, and $\frac{\partial Loss}{\partial \hat{v}}$. We derive the aforementioned derivative functions as follows

$$\begin{aligned}\frac{\partial Loss}{\partial \mu_{BN}} &= \frac{\partial Loss}{\partial \hat{v}}\frac{\partial \hat{v}}{\partial \mu_{BN}} + \frac{\partial Loss}{\partial \sigma^2_{BN}}\frac{\partial \sigma^2_{BN}}{\partial \mu_{BN}} \\ &= \frac{\partial Loss}{\partial \sigma^2_{BN}} \cdot \frac{-2(v-\mu_{BN})}{M} + \frac{\partial Loss}{\partial \hat{v}}\frac{-1}{\sqrt{\sigma^2_{BN}+\epsilon}}\end{aligned} \tag{10}$$

where $\frac{\partial Loss}{\partial \sigma^2_{BN}}$ and $\frac{\partial Loss}{\partial \hat{v}}$ are given as

$$\begin{aligned}\frac{\partial Loss}{\partial \sigma^2_{BN}} &= \frac{\partial Loss}{\partial BN}\frac{\partial BN}{\partial \hat{v}}\frac{\partial \hat{v}}{\partial \sigma^2_{BN}} \\ &= \frac{-1}{2}\frac{\partial Loss}{\partial \hat{v}} \cdot (\sigma^2_{BN}+\epsilon)^{-\frac{3}{2}} \cdot (v-\mu_{BN})\end{aligned} \tag{11}$$

$$\frac{\partial Loss}{\partial \hat{v}} = \frac{\partial Loss}{\partial BN}\frac{\partial BN}{\partial \hat{v}} = \gamma \cdot \rho \cdot sgn(fmap) \cdot (\hat{y}-y) \tag{12}$$

The derivation of loss function with respect to the input can be then be computed as follows

$$\begin{aligned}\frac{\partial Loss}{\partial v} &= \frac{\partial Loss}{\partial \mu_{BN}}\frac{\partial \mu_{BN}}{\partial v} + \frac{\partial Loss}{\partial \sigma^2_{BN}}\frac{\partial \sigma^2_{BN}}{\partial v} + \frac{\partial Loss}{\partial \hat{v}}\frac{\partial \hat{v}}{\partial v} \\ &= \frac{\partial Loss}{\partial \mu_{BN}} \cdot \frac{1}{M} + \frac{\partial Loss}{\partial \sigma^2_{BN}} \cdot \frac{2(v-\mu_{BN})}{M} \\ &+ \frac{\partial Loss}{\partial \hat{v}} \cdot \frac{1}{\sqrt{\sigma^2_{BN}+\epsilon}}\end{aligned} \tag{13}$$

When considering a FL system, it is assumed that the gradient for each layer are available. These gradients can be accessed by a malicious server to recreate the data, accordingly. For instance, whether it is a deep neural network or a shallow learning method, Equation 9 holds the information related to the data as well as BN's hyperparameter that is correlated to the feature maps extracted from input data. The correlation of the γ with the feature map can be observed in the Equations 10 and 11, respectively. We also show that the

information regarding the input data is available in the output shown in Equation 13, which can be used to initiate an attack using model inversions. From the above computations, we can conclude that for any CNN, if the gradients for BN layer, convolutional layer, and fully connected layer, are known to the attacker, it is possible for the attacker to backtrack the gradients for recreating the data and associated labels.

B. PROPOSED MODEL INVERSION DEFENSE

We propose a block encryption layer network (BELANet) that introduces the BELA module, specifically with the batch normalization layer, which uses a trainable encryption layer to prevent model inversion in FL settings. We show that our proposal is simple yet effective and does not hinder the training process since the module is twice differentiable, which is one of the requirements to train the convolutional neural network. We consider the same convolutional neural network, i.e., an input layer, a convolutional layer ($v = \vartheta \otimes X + b_v$), the feature maps of the convolutional layer ($fmap = \alpha \cdot v + \lambda$), the activation function that extracts the feature map ($Act = ReLU(fmap)$), flatten layer ($Act' = flatten(Act)$), FC layer ($FC = \rho \cdot Act' + b_{FC}$), softmax layer ($\hat{y} = softmax(FC)$), and the cross entropy loss function ($Loss = CE(\hat{y}, y)$). We add our BELA module before the extraction of feature maps from the first convolutional layer. The hyperparameters α and λ are defined as $W_\alpha \cdot RS + b_\alpha$ and $W_\lambda \cdot RS + b_\lambda$, respectively. The notation RS refers to the random sequence, W_α and W_λ are the trainable weights, while b_α and b_λ are their respective biases. The partial derivatives with respect to α and λ for the loss function are given below.

$$\frac{\partial Loss}{\partial \lambda} = (\hat{y} - y) \cdot RS \cdot rho \cdot sgn(Act) \tag{14}$$

$$\frac{\partial Loss}{\partial \alpha} = (\hat{y} - y) \cdot RS \cdot \hat{v} \cdot \rho \cdot sgn(Act) \tag{15}$$

Since the proposed method is based on zero-trust principle, the RS and associated weights with the BELA module are to be kept private, i.e. not shared with the server, which would help the model to defend itself against model inversion attacks. To prove its efficacy, we compute the gradient of the loss with respect to $\hat{v}$, which results in

$$\begin{aligned}\frac{\partial Loss}{\partial \hat{v}} &= \frac{\partial Loss}{\partial fmap}\frac{\partial fmap}{\partial \hat{v}} \\ &= \frac{\partial Loss}{\partial fmap} \cdot (\alpha \cdot RS + b_\alpha) \cdot \rho \cdot sgn(fmap) \cdot (\hat{y} - y)\end{aligned} \tag{16}$$

As the RS and associated weights are private, the parameters α and λ cannot be computed and subsequently, equations 10, 11, and 13 cannot be solved. However, equation 12 can be solved, but the attacker would be unable to determine $fmap = \alpha \cdot v + \lambda$, as the parameters cannot be determined through gradients. Based on the above computations, we show that by employing the BELA module in the CNN, the attacker would be unable to get the feature maps, thus, by extension, cannot recreate the private data.

C. BELA MODULE

In this work, we propose the BELA module. The block is assumed to be implementable in any CNN architecture. We have two trainable parameters in the BELA module, i.e., α and λ. Both parameters yield a dimension size similar to the number of channels in the convolutional layer denoted by D. The BELA module is placed as a camouflage to the BN layer; therefore, the two aforementioned parameters mimic the shift and scale transformational changes. The encryption in the BELA module is defined by the random generated sequence (RS) having length K, therefore, the weight of the BELA module yields the dimension $\mathbb{R}^{K \times D}$. Based on the aforementioned information, we can define the trainable parameters as $\alpha = W_\alpha \cdot RS + b_\alpha$ and $\lambda = W_\lambda \cdot RS + b_\lambda$. Subsequently, the embedding function $\mathbb{E}mb$ for the input and BELA module can be defined as

$$\mathbb{E}mb(X, RS) = \alpha \cdot (\vartheta \otimes X) + \lambda \tag{17}$$

As the proposed BELA module is designed for the FL settings, it is therefore necessary to explain its integration. In this regard, we provide the steps for local training and model aggregation as follows.

- The server initializes the BELA module and distributes the models to the client.
- Each client generates its own RS and trains the network locally.
- The server aggregates the models from the client while excluding the BELA module gradients.
- The global model is then sent to the clients.
- The client updates the global model locally while freezing the BELA module.

The process continues in a similar fashion while looping through the aforementioned steps. It should be noted that during the whole process the RS and the parameters are secured with the client locally, i.e. they never leave the owner. Therefore, the proposed BELA module prevents the gradients from model inversion attacks.

V. EXPERIMENTAL SETUP AND ANALYSIS

This section presents details on the datasets used to evaluate the efficacy of the proposed approach. We present two sets of experiments. The first set employs various CNN network architectures to demonstrate that the BELA module can be integrated without compromising performance. Furthermore, the first set of experiments also provides an experimental analysis of the defense against model inversion attacks by applying DLG [5], GGL [13], and SPIN [9].

The second set of experiments compares the proposed BELANet with existing defense strategies such as Soteria [10], GradDefense [11], and Outpost [26], against DLG and csDLG [6] attack methods. For the second experiment, we used the PLATO[1] framework to evaluate the efficacy of the proposed approach against gradient leakage attacks in FL

[1] https://github.com/TL-System/plato

TABLE 2. The accuracies are reported on CIFAR 10, CIFAR 100, and MNIST trained with VGG-16, ResNet-20, ResNet-34, and LeNet architecture. The best results are highlighted with bold and underline face while the second-best results are highlighted with bold and Italic face.

Dataset		CIFAR 10			CIFAR 100			MNIST
Model		VGG-16	ResNet-20	ResNet-34	VGG-16	ResNet-20	ResNet-34	LeNet
	BELA	(224 * 224)	(32 * 32)	(224 * 224)	(224 * 224)	(32 * 32)	(224 * 224)	(32 * 32)
Centralized Learning	×	88.62	90.34	91.15	62.92	66.70	72.91	**97.63**
Centralized Learning	✓	90.68	89.05	**93.12**	73.22	66.87	**77.90**	***97.60***
Federated Learning	×	87.27	90.37	88.50	60.30	57.91	67.59	97.25
Federated Learning	✓	92.27	87.29	***92.66***	72.61	60.17	***75.91***	97.19

settings. For Soteria, GradDefense, and Outpost, we use the same settings as proposed in their studies. We adopt the attack settings as proposed in [26], where the attack methods are trained for 3K iterations using the L-BFGS optimizer.

A. DATASETS

For the evaluation of the proposed work using first set of experiments, we employ five benchmark datasets, i.e. CIFAR-10 [27], CIFAR-100 [27], MNIST [28], and ILSVRC [29]. For the second set of experiments, we evaluate the proposed approach and perform a comparative analysis on the CIFAR-10 [27] and EMNIST [40], datasets. The reason for choosing the previously mentioned datasets is twofold. First, these datasets have been widely adopted in similar studies. Second, their varying complexity and image resolutions provide a sense of generalization and promote wider adoption of the proposed method. We evaluate the experimental results using the Similarity Index Measure (SSIM) [30], the Learned Perceptual Image Patch Similarity (LPIPS) [31], the Peak Signal-to-Noise Ratio (PSNR), and the Mean Square Error (MSE). In order to directly match the true data and the reconstructed data, we set the batch size to 1. The metrics LPIPS and MSE measure the similarity between two patches and the pixel-wise L2 difference between the true and reconstructed data, respectively. A lower value for both of them indicates that the similarity and performance are high. SSIM measures the structural similarity between the true and reconstructed image, while PSNR assesses the quality of the reconstructed data against the true data using the logarithmic scale. We employ ResNet34 [32] and VGG16 [33] to measure the LPIPS score. The RS length is set to 1024. The implementation of the proposed method is performed using the PyTorch framework [34]. We employ ResNet20 for CIFAR100, and ResNet34 on ILSVRC2012. The output dimension of ResNet20 and ResNet34 is 16 and 64, respectively.

B. PERFORMANCE ASSESSMENT ON ACCURACY

We assess the performance with respect to accuracy using four scenarios. The first is with a centralized training strategy without using the BELA module. The second is with a centralized training strategy with the BELA module. The third is the FL straining strategy without the BELA module, and the last is the FL strategy with the BELA module. For the aggregation process, we employed the Federated Averaging (FedAvg) method. We employ ResNet [32], VGGNet [33], and LeNet [35], as our base architectures to assess performance against the classification task. We experimented with seven configurations and four scenarios, while training a total of 28 CNN models to provide the aforementioned performance assessment. We report the experimental results in Table 1.

We have observed that the centralized learning scheme achieves the best results for each of the datasets. However, if the centralized learning is equipped with the BELA module, it achieves the best results on CIFAR 10 and CIFAR 100 while achieving the second best result on MNIST. We have also observed that the FL training scheme equipped with the BELA module achieves the second best result on CIFAR 10 and CIFAR 100 datasets. However, on MNIST, the difference in accuracy between the best results and FL scheme with BELA module is 0.44%. The results show that the BELA module does not decay the performance in terms of accuracy in the recognition tasks. The improvement with BELA in the FL scheme in CIFAR 10 was observed when using VGG-16 and ResNet-34 to be 5. 0%, 4. 2%, respectively. Similarly, improvement with BELA was observed in the FL scheme in CIFAR 100 when using VGG-16, ResNet-20, and ResNet-34 as 12. 31%, 2. 26%, and 8. 32%, respectively. The only time performance was degraded with the BELA module was in CIFAR 10 when trained with ResNet-20. The decline in accuracy was noted to be 3.08%. Overall, it can be said that the BELA module contributes to the improvement in performance, especially when the resolution is increased to 224*244, accordingly.

Table 1 provides an average accuracy computed from all the local clients. In the BELA module, every client creates a personalized model when trained with their own unique RS and BELA module parameters. In this regard, we select an optimal model from each client from over all training rounds to show the accuracy. We used a total of 3 clients in our experiment. Furthermore, we also add the accuracy results from a model that is initialized with random weights to the BELA module along with a randomly generated sequence. The results of this experiment are shown in Table 2. The results show that the LeNet on the MNIST dataset yields the largest performance gap of approximately 87% while the

TABLE 3. Client-wise reported accuracies for FL scheme with BELA module

Dataset	CIFAR 10			CIFAR 100			MNIST
Model	VGG-16	ResNet-20	ResNet-34	VGG-16	ResNet-20	ResNet-34	LeNet
Random	92.46	78.45	91.27	71.08	27.97	68.57	10.66
Clients							
Client 1	92.40	88.26	92.23	71.74	61.92	76.30	97.43
Client 2	92.78	86.42	92.63	72.57	61.28	76.17	97.45
Client 3	93.27	87.83	92.78	72.61	61.71	76.28	97.21

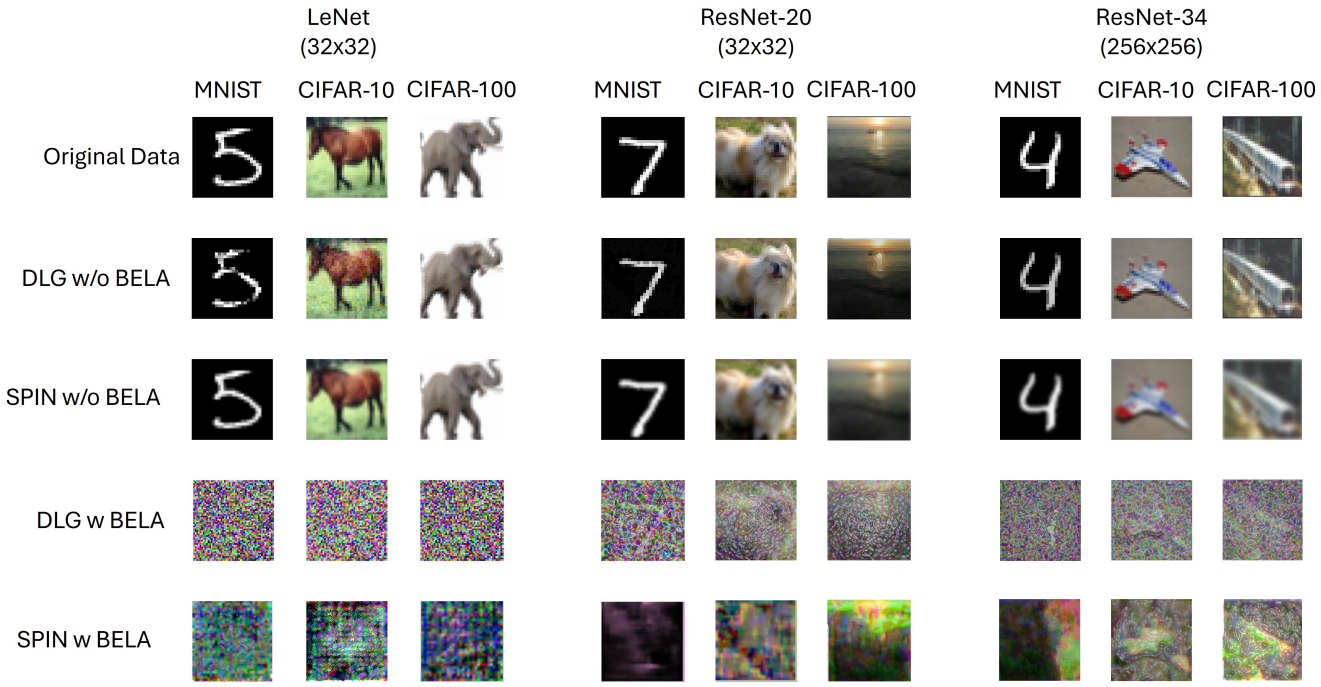

FIGURE 4. Qualitative comparison of Model Inversion Attack for Image Reconstruction with and without BELA module on MNIST, CIFAR-10, and CIFAR-100 with varying network architectures.

lowest performance gap of 0. 81% was observed on VGG-16 trained using CIFAR 10. We believe that the largest gap in LeNet is directly proportional to the size of the network as well as the number of parameters, as the LeNet is the smallest network amongst all.

C. DEFENSE AGAINST MODEL INVERSION ATTACKS

As mentioned earlier, we employed three state-of-the-art attack methods, i.e., DLG [5], GGL [13], and SPIN [9] to evaluate the efficacy of the proposed BELA module for the defense mechanism. The reason for choosing the aforementioned methods is their strong capability of performing model inversion attacks as well as their implementation, which is publicly available. The resolution and batch size of 32×32 and 1 were used for CIFAR-100, CIFAR-10, and MNIST when trained with LeNet and ResNet20. For ResNet34, we considered the resolution of 256×256 to comply with the SPIN method. The qualitative results against the aforementioned model inversion attacks and for the proposed BELA module are shown in Fig. 4.

The results clearly demonstrate that DLG and SPIN, both are capable of reconstructing the data on MNIST, CIFAR-10, and CIFAR-100. It is also observed that the reconstruction is better with larger-resolution data compared to smaller-resolution data. The results with the BELA module are also shown in Fig. 4. The DLG method mainly targets the feature space, but due to the use of the BELA module, the DLG cannot reconstruct the data on either of the datasets with varying architectures. We can see some patterns emerging when using SPIN with larger resolution data; however, as will be shown in the quantitative analysis, the pattern emergence does not help in data reconstruction, accordingly. We also present the quantitative analysis for the model inversion attacks with and without the BELA module in Table 3. We evaluated the efficacy of the proposed method with respect to SSIM, LPIPS, PSNR, and MSE. The value of the evaluation metrics are computed between the true data and the reconstructed data. The results provide consistent traits when using different methods, datasets, or network architectures. These traits include lower similarity when using the BELA module, lower LPIPS values, lower PSNR, and higher MSE. For instance, the MSE when using DLG using

TABLE 4. Comparative Analysis of the defense against model inversion attacks using DLG, SPIN and GGL with and without BELA module. The metric computation is carried out using the true and reconstructed data.

Method	Dataset	Model	SSIM		LPIPS-A		LPIPS-V		PSNR		MSE	
			w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/
DLG	**MNIST**	LeNet (32*32)	0.719	0.002	0.11	0.53	0.25	0.67	32.65	25.49	50.04	143.76
DLG		ResNet-20 (32*32)	0.996	0.007	0.01	0.32	0.06	0.62	46.95	27.78	1.54	102.34
SPIN		LeNet (32*32)	0.998	0.107	0.01	0.43	0.01	0.68	51.94	27.56	0.34	104.08
SPIN		ResNet-20 (32*32)	0.996	0.008	0.01	0.35	0.03	0.63	47.05	27.98	1.59	101.97
DLG	**CIFAR-10**	LeNet (32*32)	0.718	0.001	0.06	0.29	0.16	0.52	40.87	27.75	30.14	99.39
DLG		ResNet-20 (32*32)	0.964	0.046	0.01	0.27	0.02	0.47	40.38	29.14	9.43	85.88
DLG		ResNet-34 (256*256)	0.946	0.139	0.05	0.61	0.03	0.49	37.94	28.41	14.55	71.59
SPIN		LeNet (32*32)	1.00	0.100	0.01	0.25	0.01	0.48	48.52	28.29	0.75	91.78
SPIN		ResNet-20 (32*32)	0.982	0.049	0.01	0.23	0.01	0.42	39.85	27.86	8.42	86.12
SPIN		ResNet-34 (256*256)	0.953	0.165	0.02	0.52	0.02	0.45	37.56	29.41	15.55	71.59
DLG	**CIFAR-100**	LeNet (32*32)	0.858	0.007	0.05	0.35	0.07	0.47	39.09	27.85	24.44	101.30
DLG		ResNet-20 (32*32)	0.998	0.082	0.02	0.25	0.02	0.45	47.52	27.29	1.78	100.65
DLG		ResNet-34 (256*256)	0.913	0.154	0.04	0.51	0.05	0.51	37.28	28.09	33.55	84.59
SPIN		LeNet (32*32)	0.998	0.084	0.02	0.27	0.02	0.48	47.52	27.65	1.03	100.12
SPIN		ResNet-20 (32*32)	0.962	0.031	0.02	0.23	0.02	0.44	37.85	27.86	20.42	98.79
SPIN		ResNet-34 (256*256)	0.915	0.157	0.03	0.54	0.04	0.47	33.56	28.13	34.55	84.59
DLG	**ILSVRC**	ResNet-34 (256*256)	0.928	0.268	0.04	0.45	0.06	0.43	37.97	29.36	13.33	63.36
SPIN		ResNet-34 (256*256)	0.931	0.272	0.03	0.47	0.05	0.39	38.22	29.36	13.87	63.29
GGL		ResNet-34 (256*256)	0.229	0.174	0.38	0.45	0.39	0.46	29.79	28.22	63.22	86.59

ResNet-20 without BELA module is 1.54, however with the BELA module, the MSE is increased to 102.34, therefore the percentage difference yields around 194%. Similarly to the DLG method, SPIN and GGL yield similar results for all datasets and network architectures. From the results, it was observed that the BELA module is quite effective with DLG and SPIN. As the GGL method generates true data with respect to structure and semantics, the similarity results are better for GGL; however, it can be observed that the GGL method's reconstruction capability is still restricted when using the BELA module. From the aforementioned results, we can deduce that the BELA module is quite capable of providing defense against model inversion attacks.

D. Comparison with existing Defense methods

We evaluate our proposed defense method and compare it with existing approaches using the CIFAR-10 and EMNIST datasets. Specifically, we compare our approach with the Soteria, GradDefense, and Outpost methods. Our evaluation is conducted with only one client participating, and the attack is carried out in the first round of training. To make the attack stronger, we did not apply the learning rate scheduler, weight

TABLE 5. Characteristics of employed scenarios for the validation of defense strategies.

Scenario	Attack	Dataset	BatchSize	Samples	Epoch
1	DLG csDLG	EMNIST	1	1	1
2	DLG csDLG	EMNIST	1	2	1
3	DLG csDLG	CIFAR-10	1	1	1
4	DLG csDLG	CIFAR-10	16	16	1

decay, or momentum in the SGD optimizer. The results presented are the average of 10 trials with the worst defense results, respectively. This evaluation is in line with existing methods [26]. We evaluated our method and compared it with the existing one in four scenarios for each attack. The characteristics of each scenario are defined in Table 5. We present the results for each scenario using the Soteria, GradDefense, Outpost, and proposed approaches, measured by SSIM, LPIPS, and MSE, respectively, in Table 6. Overall, the csDLG attack method yields better reconstruction results compared to the DLG method. It can be seen that MSE is the most inconsistent evaluation metric, in accordance with [26]. However, the proposed BELA achieves the best and second-best results in terms of LPIPS and SSIM. The results demonstrate that BELANet provides the most generalized defense across all scenarios, datasets, attack methods, and evaluation metrics.

VI. CONCLUSION

The zero-trust systems are getting popular for providing defense against variety of attacks in the next-generation communication systems. FL systems provide data security, as the data never leave the owner; however, the same cannot be said for the model parameters. FL systems can benefit from zero-trust systems by implementing a defense mechanism which does not trust the server by keeping some of the parameters private in order to restrain the attacker from reconstructing data. We propose the Transformational Block Encryptor (BELA) module in FL setting that is based on a zero-trust system. The BELA module mimics the BN layer in the CNN network architecture. The parameters for the BELA module are optimized during training, while the parameter values are kept private. Each of the clients have their own parameters, so even though the attacker get their hands on any of the client's parameters, it is difficult to reconstruct the data from a global model. We provide theoretical analysis for the model inversion attack using deep neural networks and convolutional neural networks. We also provide a detailed analysis on performance assessment and the defense mechanism against model inversion attacks. The results prove that the BELA module is quite effective in providing defense against model inversion attacks in FL setting while not compromising on the accuracy.

As future research, we plan to test the BELA module in a Federated Learning-based testbed on communication-related data, such as channel coding and optimization, to assess its integrity in vehicular networks. Additionally, we aim to enhance the BELA module with quantum encryption to further secure the trained model from model inversion attacks.

REFERENCES

[1] K. Dev, S. A. Khowaja, P. K. Sharma, B. S. Chowdhry, S. Tanwar and G. Fortino, "DDI: A Novel Architecture for Joint Active User Detection and IoT Device Identification in Grant-Free NOMA Systems for 6G and Beyond Networks," in IEEE Internet of Things Journal, vol. 9, no. 4, pp. 2906-2917, 15 Feb.15, 2022, doi: 10.1109/JIOT.2021.3095255.

[2] S. A. Khowaja, K. Dev, P. Khowaja and P. Bellavista, "Toward Energy-Efficient Distributed Federated Learning for 6G Networks," in IEEE Wireless Communications, vol. 28, no. 6, pp. 34-40, December 2021, doi: 10.1109/MWC.012.2100153.

[3] M. Zawish et al., "AI and 6G Into the Metaverse: Fundamentals, Challenges and Future Research Trends," in IEEE Open Journal of the Communications Society, vol. 5, pp. 730-778, 2024, doi: 10.1109/OJ-COMS.2024.3349465.

[4] S. A. Khowaja, K. Dev, N. M. F. Qureshi, P. Khuwaja and L. Foschini, "Toward Industrial Private AI: A Two-Tier Framework for Data and Model Security," in IEEE Wireless Communications, vol. 29, no. 2, pp. 76-83, April 2022, doi: 10.1109/MWC.001.2100479.

[5] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2019, pp. 14774–14784. [Online]. Available: https://dl.acm.org/doi/10.5555/3454287.3455610

[6] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients–How easy is it to break privacy in federated learning?" in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 33, 2020, pp. 16937–16947.

[7] B. Zhao, K. R. Mopuri, and H. Bilen, "IDLG: Improved deep leakage from gradients," 2020, arXiv:2001.02610

[8] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, "See through gradients: Image batch recovery via grad-inversion," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 16337–16346.

[9] S. A. Khowaja, P. Khuwaja, K. Dev and A. Antonopoulos, "SPIN: Simulated Poisoning and Inversion Network for Federated Learning-Based 6G Vehicular Networks," ICC 2023 - IEEE International Conference on Communications, Rome, Italy, 2023, pp. 6205-6210, doi: 10.1109/ICC45041.2023.10279339.

[10] J. Sun, A. Li, B. Wang, H. Yang, H. Li, and Y. Chen, "Soteria: Provable defense against privacy leakage in federated learning from representation perspective," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 9311–9319.

[11] J. Wang, S. Guo, X. Xie, and H. Qi, "Protect privacy from gradient leakage attack in federated learning," in Proc. INFOCOM IEEE Conf. Comput. Commun., May 2022, pp. 580–589.

[12] S. A. Khowaja, I. H. Lee, K. Dev, M. A. Jarwar and N. M. F. Qureshi, "Get Your Foes Fooled: Proximal Gradient Split Learning for Defense Against Model Inversion Attacks on IoMT Data," in IEEE Transactions on Network Science and Engineering, vol. 10, no. 5, pp. 2607-2616, 1 Sept.-Oct. 2023, doi: 10.1109/TNSE.2022.3188575.

[13] Z. Li, J. Zhang, L. Liu, and J. Liu, "Auditing privacy defenses in federated learning via generative gradient leakage," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 10132–10142.

[14] D. Scheliga, P. Mader, and M. Seeland, "Precode-a generic model extension to prevent deep gradient leakage," in Winter Conference on Applications of Computer Vision, 2022, pp. 1849–1858.

[15] H. Lee, J. Kim, S. Ahn, R. Hussain, S. Cho, and J. Son, "Digestive neural networks: A novel defense strategy against inference attacks in federated learning," computers & security, vol. 109, p. 102378, 2021.

[16] S. A. Khowaja, L. Nkenyereye, P. Khowaja, K. Dev and D. Niyato, "SLIP: Self-Supervised Learning Based Model Inversion and Poisoning

TABLE 6. Comparative Analysis of Defense Strategies with the Proposed BELA module using varying configurations. The bold, underline combination represents the best results for each scenario and evaluation metric while the italic, underline combination represent the second-best results.

Scenario	Method	csDLG			DLG		
		SSIM	LPIPS	MSE	SSIM	LPIPS	MSE
1	No Defense	1.00	5.8e-7	2.6e-7	0.99	7.1e-2	6.6e-3
	Outpost	1.6e-2	*0.68*	294.68	0.13	0.58	77.05
	GradDefense	1.7e-2	0.64	**360.98**	1.0e-2	**0.64**	32.57
	Soteria	**4.1e-2**	0.63	296.76	**3.3e-2**	*0.63*	**95.19**
	BELA	*3.8e-2*	**0.71**	*348.61*	*2.7e-2*	*0.63*	*89.46*
2	No Defense	0.70	0.40	0.14	0.77	0.31	5.8e-2
	Outpost	*3.8e-2*	*0.66*	277.73	*6.4e-2*	0.67	*75.92*
	GradDefense	3.4e-2	0.63	*668.79*	2.7e-2	0.65	25.50
	Soteria	1.6e-2	0.59	**782.93**	4.7e-2	*0.70*	16.07
	BELA	**4.2e-2**	**0.70**	436.87	**6.7e-2**	**0.72**	**83.44**
3	No Defense	0.99	1.8e-3	5.9e-5	0.57	0.53	5.1e-2
	Outpost	3.4e-2	*0.77*	35.24	2.1e-2	*0.77*	13.10
	GradDefense	1.7e-2	*0.77*	**56.66**	*2.2e-2*	0.74	11.46
	Soteria	**5.5e-2**	0.76	25.91	**5.5e-2**	0.76	**25.91**
	BELA	*4.1e-2*	**0.79**	*48.62*	**5.5e-2**	**0.78**	*18.56*
4	No Defense	0.30	0.51	6.2e-2	2.9e-2	0.67	1.37
	Outpost	3.1e-2	0.70	13.03	1.7e-2	*0.70*	*15.85*
	GradDefense	*3.8e-2*	*0.71*	**2.1e10**	*2.2e-2*	0.69	4.04
	Soteria	2.7e-2	0.69	14.87	*2.2e-2*	*0.70*	2.85
	BELA	**4.1e-2**	**0.73**	*20.54*	**2.4e-2**	**0.74**	**18.43**

Detection-Based Zero-Trust Systems for Vehicular Networks," in IEEE Wireless Communications, vol. 31, no. 2, pp. 50-57, April 2024, doi: 10.1109/MWC.001.2300377.

[17] W. Wei, L. Liu, Y. Wut, G. Su, and A. Iyengar, "Gradient-leakage resilient federated learning," in 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS). IEEE, 2021, pp. 797–807

[18] W. Wei, L. Liu, M. Loper, K.-H. Chow, M. E. Gursoy, S. Truex, and Y. Wu, "A framework for evaluating gradient leakage attacks in federated learning," arXiv:2004.10397, 2020

[19] L. Fan, K. W. Ng, C. S. Chan, and Q. Yang, "Deepip: Deep neural network intellectual property protection with passports," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.

[20] S. A. Khowaja, P. Khuwaja, K. Dev, A. Antonopoulos and M. Magarini, "DASTAN-CNN: RF Fingerprinting for the Mitigation of Membership Inference Attacks in 5G," GLOBECOM 2023 - 2023 IEEE Global Communications Conference, Kuala Lumpur, Malaysia, 2023, pp. 5524-5529, doi: 10.1109/GLOBECOM54140.2023.10437263.

[21] J. Jeon, K. Lee, S. Oh, J. Ok et al., "Gradient inversion with generative image prior," Advances in Neural Information Processing Systems, vol. 34, 2021.

[22] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in ACM CCS, 2017, pp. 603–618.

[23] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in IEEE INFOCOM 2019-IEEE Conference on Computer Communications. IEEE, 2019, pp. 2512–2520.

[24] S. A. Khowaja, P. Khuwaja, K. Dev, K. Singh, L. Nkenyereye, and D. Kilper, "ZETA: ZEro-Trust Attack Framework with Split Learning for Autonomous Vehicles in 6G Networks," in IEEE Wireless Communications and Networking Conference Workshops, 2024, doi:10.1109/WCNC57260.2024.10571158.

[25] M. A. P. Chamikara, P. Bertok, I. Khalil, D. Liu, and S. Camtepe, "Privacy preserving distributed machine learning with federated learning," Computer Communications, vol. 171, pp. 112–125, 2021.

[26] F. Wang, E. Hugh and B. Li, "More Than Enough Is Too Much: Adaptive Defenses Against Gradient Leakage in Production Federated Learning," in IEEE/ACM Transactions on Networking, doi: 10.1109/TNET.2024.3377655.

[27] A. Krizhevsky, G. Hinton et al., "Learning multiple layers of features from tiny images," Citeseer, 2009.

[28] Y. LeCun, "The mnist database of handwritten digits," http://yann. lecun. com/exdb/mnist/, 1998.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009, pp. 248–255.

[30] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," IEEE transactions on image processing, vol. 13, no. 4, pp. 600–612, 2004.

[31] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 586–595.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in IEEE CVPR, 2016, pp. 770–778.

[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv:1409.1556, 2014.

[34] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga et al., "Pytorch: An imperative style, high performance deep learning library," Neural Information Processing Systems, vol. 32, pp. 8026–8037, 2019

[35] 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.

[36] X. Lin, "An Overview of 5G Advanced Evolution in 3GPP Release 18," in IEEE Communications Standards Magazine, vol. 6, no. 3, pp. 77-83, September 2022.

[37] J. Hoydis, F. A. Aoudia, A. Valcarce and H. Viswanathan, "Toward a 6G AI-Native Air Interface," in IEEE Communications Magazine, vol. 59, no. 5, pp. 76-81, May 2021.

[38] Y. Sun, Z. Liu, J. Cui, J. Liu, K. Ma and J. Liu, "Client-Side Gradient Inversion Attack in Federated Learning Using Secure Aggregation," in IEEE Internet of Things Journal, vol. 11, no. 17, pp. 28774-28786, 1 Sept.1, 2024, doi: 10.1109/JIOT.2024.3405939.

[39] Kim J, Park S. Random Gradient Masking as a Defensive Measure to Deep Leakage in Federated Learning. arXiv preprint arXiv:2408.08430. 2024 Aug 15.

[40] G. Cohen, S. Afshar, J. Tapson and A. van Schaik, "EMNIST: Extending MNIST to handwritten letters," 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 2017, pp. 2921-2926, doi: 10.1109/IJCNN.2017.7966217.