Défense contre les attaques par porte dérobée en apprentissage fédéré par estimation du motif d'attaque et élagage

Fabiola ESPINOZA CASTELLON[1] Deepika SINGH[1] Aurélien MAYOUE[1] Cédric GOUY-PAILLER[1]

[1]Institut LIST, CEA, Université Paris-Saclay, F-91120, Palaiseau, France

Résumé – Dans un cadre fédéré, nous proposons une défense contre les attaques par porte dérobée déclenchées par l'ajout d'un motif au sein des données d'entraînement. Notre défense s'appuie sur les paramètres du modèle fédéré pour estimer le motif déclencheur et ensuite élaguer les neurones responsables de l'attaque au sein du réseau. Contrairement aux approches existantes, notre approche ne nécessite pas d'informations supplémentaires de la part des participants au processus d'apprentissage collaboratif et surtout s'avère robuste face à l'hétérogénéité des données et ce quel que soit le nombre d'acteurs malveillants. En comparaison avec les méthodes de l'état-de-l'art, notre défense apparaît plus adaptée pour atténuer les attaques par porte dérobée tout en préservant une grande précision sur des données bénignes.

Abstract – This paper proposes a post-training defense against pattern-triggered backdoor attacks in federated learning contexts. This approach relies first on the server estimating the attack pattern. The server then prunes parameters of the attacked model that are responsible for the attack. This scheme offers an improvement over the existing approaches by demonstrating robustness to data heterogeneity among users without needing additional information from users and regardless of the number of malicious clients. Based on comparison with existing state-of-the-art methods, the proposed method is shown to succeed in mitigating backdoor attacks while preserving high accuracy on clean inputs.

1 Introduction

L'apprentissage fédéré (AF) [5] est un nouveau paradigme d'apprentissage automatique qui permet à plusieurs entités, que nous appelons clients, d'entraîner un modèle de façon collaborative mais sans jamais partager leurs données. Ce processus est coordonné par un serveur central et nécessite plusieurs tours d'apprentissage pour réaliser l'entraînement du modèle fédéré. Ainsi, à chaque tour, le serveur transmet le modèle courant aux clients qui vont alors mettre à jour ses paramètres en l'entraînant indépendamment à partir de leurs propres données pour ensuite les retourner au serveur qui va les agréger. Dans sa version initiale, l'AF se base sur FedAvg [5], qui utilise la moyenne pondérée comme règle d'agrégation. Au tour T, les paramètres calculés par le serveur sont donc $w_T = \sum_{k \in C_T} \frac{n_k}{N} w_T^k$, où C_T est l'ensemble des clients participant au tour T, w_T^k sont les paramètres envoyés par le client k, n_k est le nombre de donnés du client k et $N = \sum_{q \in C_t} n_q$.

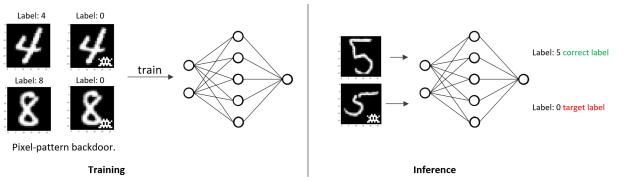

FIGURE 1 : Principe d'une attaque par porte dérobée

Du point de vue serveur, n'ayant pas accès aux données clients, il n'a aucun moyen de vérifier que les paramètres reçus à chaque tour correspondent bien à un entraînement sur des données licites. L'AF est ainsi vulnérable à la présence de clients malveillants qui souhaiteraient dégrader les performances du modèle [1]. Nous nous intéressons ici aux attaques "par porte dérobée" [3] que l'on nommera BA ("backdoor attacks"). Pendant la phase d'entraînement, un groupe de clients malveillants ajoute un motif d'attaque, également nommé déclencheur, à leurs données et leur associe une classe cible t. Lors de l'inférence, le modèle va ainsi incorrectement prédire la classe cible t pour les données empoisonnées tandis que les données bénignes seront correctement classées (Fig 1) ce qui rend les BA difficilement détectables. Nous proposons ici une défense pour contrer les BA adaptée à un modèle de type réseau de neurones et qui consiste à estimer le motif déclencheur pour ensuite élaguer les neurones responsables de l'attaque.

2 Travaux antérieurs

En AF, les premières défenses ont consisté à modifier la règle d'agrégation FedAvg dans le but d'écarter les contributions aberrantes dans leur globalité avec Krum [1] ou plus finement en appliquant la médiane par coordonnée (Med) [9]. Dans la même mentalité, Ozdayi et al. [6] proposent RLR qui ajuste le taux d'apprentissage par coordonnée et par tour afin d'atténuer les informations contradictoires des clients. Sun et al. [7] quant à eux appliquent une faible confidentialité différentielle (fCD) au modèle fédéré en amont de l'agrégation. Ces défenses s'avèrent efficaces lorsque la distribution des données entre les clients est indépendante et identiquement distribuée (IID) ce qui est rarement le cas en fédéré [4]. Et lorsque le degré d'hétérogénéité des données augmente, la robustesse de ces méthodes se dégrade par le fait que les poids des clients divergent sans nécessairement être malveillants.

Les défenses qui adressent le challenge de l'hétérogénéité des données nécessitent des informations supplémentaires envoyées par les clients. Zhao et al. [10] proposent FedReverse pour reconstruire le motif d'attaque par agrégation à partir des

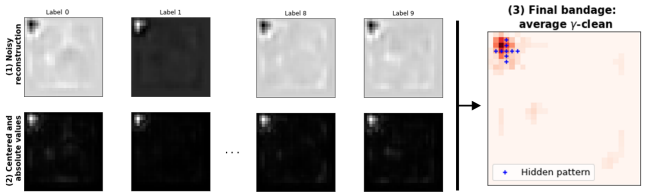

FIGURE 2 : Procédé de reconstruction du motif déclencheur (croix en haut à gauche)

estimations réalisées par les clients. Wu et al. [8] proposent une défense qui identifie les neurones "dormants", i.e. les neurones à faible activation sur les données bénignes mais responsables de la BA. Le serveur élimine les neurones par un processus de vote majoritaire basé sur les cartes d'activation envoyées par les clients. Cependant, le serveur reste aveugle quant à la fiabilité des informations qu'il reçoit de la part des clients et ces deux défenses peuvent donc être mises en échec si une majorité des clients envoient des informations erronées.

Dans ce papier, nous proposons une méthode qui répond aux deux principales faiblesses des approches de l'état-de-l'art, à savoir être robuste aux cas non-IID et ce sans s'appuyer sur des informations (potentiellement erronées) envoyées par les clients. Si bien notre méthode peut en principe s'appliquer en centralisé, en pratique, elle est plus adaptée en fédéré parce que le serveur n'utilise pas de données d'entraînement.

3 Méthode proposée

Nous proposons une défense robuste aux BA dans un contexte fédéré qui implique que les données sont hétérogènes entre les clients. Notre méthode reprend l'idée de l'élagage de neurones responsables de l'attaque mais sans se reposer sur les informations envoyées par des clients potentiellement malveillants. Pour cela nous allons reconstruire le motif déclencheur à partir des paramètres du modèle fédéré et l'utiliser pour identifier les neurones à l'origine de l'attaque puis les éliminer. Notre défense se met en place après le processus d'AF et est ainsi agnostique du nombre de clients malicieux.

3.1 Reconstruction du déclencheur d'attaque

3.1.1 Estimer le motif d'attaque

L'entraînement fédéré minimise une fonction objectif $f = \sum_i^K p_i f_i$, où f_i est la fonction objectif du client i, K le nombre total de clients et p_i la pondération du client i. Si à un tour T le modèle a été empoisonné alors les poids w_T ont été optimisés de sorte que f ait une valeur minimale pour la classe d'attaque t et pour une entrée corrompue z_t. Pour reconstruire le déclencheur d'attaque qui est fortement corrélé à la classe cible t, nous utilisons une approche d'inversion de modèle similaire à [2]. Notre objectif est de trouver une entrée $\mathbf{z}_\tau$ qui correspond à un minimum sur la fonction objective globale f associée aux poids corrompus w_T et à l'étiquette τ, que nous voulons être la classe cible t :

$$\min_{\mathbf{z}_\tau} f(\mathbf{z}_\tau, \tau; w_T) \tag{1}$$

Pour optimiser l'Éq.1, le serveur a besoin d'un accès en boîte blanche au modèle fédéré, mais également à des données

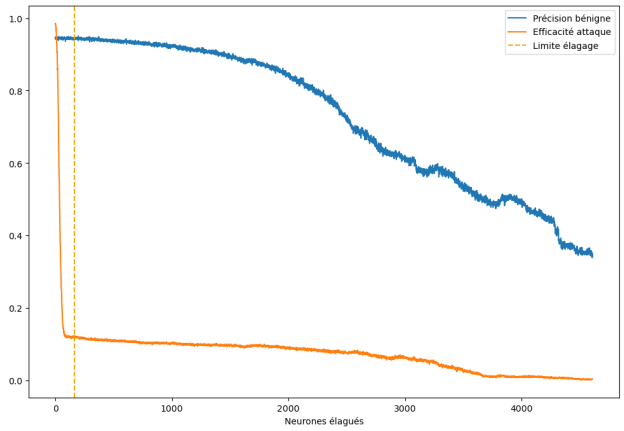

FIGURE 3 : Évolution de PB et EA lors de l'élagage.

d'entrée et à l'étiquette cible t. Cependant, dans un processus fédéré, le serveur n'a accès ni aux données empoisonnées, ni aux données propres, ni à la classe cible t. Notre approche permet de contourner ces obstacles.

Entrées de la fonction objectif : Nous initialisons $\mathbf{z}_\tau$ par un échantillon de taille n d'une distribution uniforme. De plus, nous utilisons la fonction d'entropie croisée discrète comme fonction à minimiser f (Éq.1) et une descente de gradient stochastique (SGD) comme optimiseur.

Classe cible pour l'objectif de défense : Le serveur ne connaît pas la classe cible t, mais il connaît l'ensemble des classes possibles $\mathcal{T}$ (taille de la couche de sortie). Il peut donc résoudre Éq.1 pour chaque classe. Les motifs reconstruits pour chaque classe sont visibles dans la Figure 2 (étape 1). Il apparaît que la reconstruction la plus précise est logiquement obtenue pour la classe cible t (étiquette 1 dans la Fig 2). Deux observations peuvent être faites à ce stade. Premièrement, la classe ayant la valeur la plus faible de la fonction de perte $\tilde{f}$ est communément la classe cible t. Ceci s'explique par une forte corrélation entre le motif et l'étiquette attaquée t. Deuxièmement, les reconstructions des autres classes nous fournissent également des informations sur l'attaque. En effet, comme le déclencheur est inséré sur des données d'apprentissage appartenant à l'ensemble des classes, il est logique qu'il ait un impact sur l'inversion de modèle réalisée en considérant chaque classe. Pour contrer l'effet de l'attaque, les reconstructions des classes autres que la classe cible t, produisent ainsi des valeurs opposées à l'estimation de t, et donc au déclencheur original. Bien que ces reconstructions soient plus bruitées que celle de la classe cible t, nous les utiliserons par la suite pour améliorer la reconstruction obtenue à partir du label cible.

3.1.2 Nettoyer les reconstructions

Le défenseur peut donc produire $card(\mathcal{T})$ reconstructions. Chaque reconstruction de la classe τ, notée $\mathbf{z}_\tau \in \mathbb{R}^{n \times d}$ (où d est la dimension des données d'entrée), peut être plus ou moins bruitée (Fig 2 - étape 1). Selon la classe (cible t ou non), les valeurs de notre zone d'intérêt peuvent être opposées au vrai déclencheur, mais elles représentent des valeurs extrêmes au sein de leurs reconstructions. De plus, étant donné que l'attaque concerne généralement un sous-ensemble limité de l'entrée, il suffit de conserver une quantité minimale de l'information dans $\mathbf{z}_\tau$. Nous ne gardons donc que les valeurs extrêmes qui reflètent l'effet du déclencheur caché.

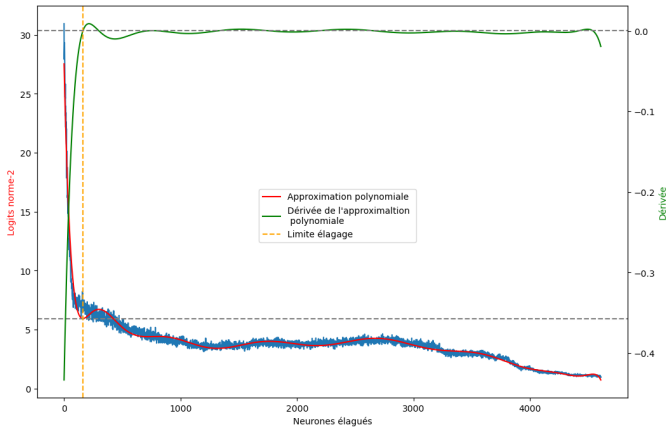

FIGURE 4 : Évolution de la différence moyenne entre logits bénins et corrompus lors de l'élagage.

Pour ceci, nous calculons la moyenne par classe des n échantillonsantillons optimisés (Fig 2 - étape 1). Pour faire ressortir les valeurs extrêmes, nous centralisons et considérons leurs valeurs absolues (Fig 2 - étape 2). Ensuite, pour rassembler nos reconstructions, nous les moyennons et décidons de mettre plus de poids sur la reconstruction associée à la fonction de perte la plus faible car elle est généralement associée à la fonction de perte de l'étiquette cible t et est donc moins bruitée. Nous obtenons une reconstruction commune $\bar{\mathbf{z}}$:

$$\bar{\mathbf{z}} = \frac{1}{2}\tilde{p}_{t_{min}} + \sum_{\substack{\tau\in\mathcal{T}\\ \tau\neq t'}} \frac{1}{2(card(\mathcal{T})-1)}\tilde{p}_\tau \tag{2}$$

où $\tilde{p}_\tau = (\tilde{p}_\tau)_{j\in[1,..,d]}$ et

$$(\tilde{p}_\tau)_j = \left| \frac{1}{n}\sum_{i=1}^{n}\left((\mathbf{z}_\tau)_{i,j} - \frac{1}{d}\sum_{j=1}^{d}(\mathbf{z}_\tau)_{i,j}\right)\right| \tag{3}$$

Enfin, nous voulons uniquement conserver la proportion γ des valeurs les plus élevées (Fig 2 - étape 3). Soit σ une permutation qui trie $(\bar{\mathbf{z}}_i)_{i\in[1,...,d]}$ par ordre croissant : $\bar{\mathbf{z}}_{\sigma(1)} \leq \bar{\mathbf{z}}_{\sigma(2)} \leq ... \leq \bar{\mathbf{z}}_{\sigma(d)}$. Nous choisissons $\bar{\mathbf{z}}_{\sigma(k)}$ tel que $k \geq d(1-\gamma)$. Notre estimation finale nettoyée $\hat{\mathbf{T}}$ du déclencheur est $\texttt{clean}(\bar{\mathbf{z}}) \in \mathbb{R}^d$:

$$\texttt{clean}(\bar{\mathbf{z}})_i = \begin{cases} \bar{\mathbf{z}}_i & \text{si } \bar{\mathbf{z}}_i \geq \bar{\mathbf{z}}_{\sigma(k)} \\ 0 & \text{sinon} \end{cases} \tag{4}$$

$\hat{\mathbf{T}}$ est ainsi une reconstruction approchant le déclencheur d'attaque (sur la Fig 2 - étape 3, le motif d'attaque est en bleu).

3.2 Élaguer le modèle pour affaiblir l'attaque

Pour simuler des données corrompues, nous ajoutons le déclencheur reconstruit à des entrées aléatoires (distributions uniformes). Ces données sont ensuite fournies au modèle pour construire une carte d'activations correspondant aux données corrompues. Nous utilisons des réseaux de neurones convolutifs pour lesquels les couches convolutives (CC) sont généralement suivies de couches connectées (FC). Concrètement, l'espace des features correspond aux activations issues des CC. Si F est le nombre de filtres de la dernière CC, et d_f est la taille des matrices d'activations issues de cette CC (feature maps), alors nous aurons $F \times d_f \times d_f$ activations. Pour contrer l'effet de la BA, nous élaguons les neurones liés aux activations ayant les valeurs les plus élevées pour le déclencheur.

Nous pouvons ainsi trier par ordre décroissant les neurones liés aux activations élevées pour le déclencheur. Nous devons également définir un nombre de neurones à élaguer ou un critère d'arrêt afin de supprimer l'effet de la BA sans (trop) dégrader les performances de la tache principale.

Supposons que nous avons accès aux données bénignes et corrompues. Nous pouvons alors élaguer les poids de façon descendante et mesurer la précision sur les données bénignes (PB), et l'efficacité de l'attaque (EA), qui est la proportion de données corrompues incorrectement prédites comme la classe cible t. Plus EA est proche de 1, plus l'attaque est efficace. Nous remarquons (Fig. 3) une forte chute de l'EA au début, et ensuite progressivement de la PB. Idéalement, nous voulons élaguer le nombre de neurones qui permet de minimiser l'attaque sans considérablement dégrader la PB.

Nous souhaitons donc que l'ajout du déclencheur ne modifie pas la prédiction d'une entrée, soit les logits de cette entrée. Soit $h_{w_T}(u_b)$ les logits d'une entrée bénigne u_b et $h_{w_T}(g(u_b))$ les logits de cette même entrée, à laquelle à été ajouté le déclencheur par la fonction g :

$$g(u_b) = \begin{cases} u_b & \text{si } \hat{\mathbf{T}} = 0 \\ \hat{\mathbf{T}} & \text{sinon} \end{cases} \tag{5}$$

Nous pouvons évaluer une différence moyenne entre des logits bénins et corrompus $d_{logits}(n) = \frac{1}{n}\sum_i \|h_{w_T}(u_i) - h_{w_T}(g(u_i))\|_2$ où u_i est un échantillonsantillon d'une distribution uniforme. Si le modèle est peu sensible à l'ajout du déclencheur, donc robuste à l'attaque, alors cette quantité est faible. En évaluant $d_{logits}(n)$ au fur et à mesure de l'élagage (Fig 4 en bleu), nous remarquons que l'élagage des premiers neurones diminue fortement $d_{logits}(n)$, et par conséquent l'effet de l'attaque. Ensuite, $d_{logits}(n)$ se stabilise, et continue à diminuer lentement. En effet, si tous les neurones sont élagués, les logits seront similaires mais ne contiendront plus d'information utile. Nous voulons donc repérer le moment où l'attaque est affaiblie mais la tâche principale n'est pas fortement impactée, donc le moment où $d_{logits}(n)$ se stabilise. Pour ceci, nous approximons $d_{logits}(n)$ par un polynôme (Fig 4 en rouge) de degré élevé (ici 15). La dérivé du polynôme (Fig 4 en vert) est négative au début de l'élagage, car notre polynôme diminue fortement. Elle s'annule ensuite parce que le polynôme oscille sur des valeurs proches. Nous fixons notre critère d'arrêt au nombre qui annule la dérivée de notre polynôme pour la première fois (Fig 4 en orange).

4 Résultats

4.1 Protocole expérimental

Pour valider notre méthode, nous avons réaliser des expériences sur MNIST et FashionMNIST. Nous simulons un contexte fédéré de 100 clients en divisant les données d'entraînement en groupes de 100 et en ajoutant trois types de motif d'attaque (carré □, croix + et copyright ©) à 20 groupes de façon à ce que 20% des clients se comportent de manière malveillante. Nous utilisons un réseau de neurones convolutifs standard constitué de deux couches CC et deux couches FC similaire à celui utilisé dans [5].

			Sans défense		Notre défense	
Dataset	IID	Motif d'attaque	PB	EA	PB	EA
MNIST	OUI	Carré □	0.95 ± 0.00	0.99 ± 0.00	0.94 ± 0.00	0.11 ± 0.01
		Croix +	0.95 ± 0.00	0.99 ± 0.00	0.94 ± 0.00	0.13 ± 0.02
		Copyright ©	0.95 ± 0.00	0.96 ± 0.00	0.88 ± 0.09	0.17 ± 0.06
	NON	Carré □	0.88 ± 0.00	0.99 ± 0.00	0.88 ± 0.00	0.15 ± 0.02
		Croix +	0.88 ± 0.00	0.99 ± 0.00	0.88 ± 0.00	0.19 ± 0.02
		Copyright ©	0.89 ± 0.00	0.98 ± 0.00	0.83 ± 0.02	0.17 ± 0.05
FashionMNIST	OUI	Carré □	0.82 ± 0.01	0.84 ± 0.03	0.81 ± 0.01	0.15 ± 0.05
		Croix +	0.82 ± 0.00	0.96 ± 0.01	0.82 ± 0.01	0.26 ± 0.06
		Copyright ©	0.83 ± 0.01	0.82 ± 0.04	0.81 ± 0.01	0.36 ± 0.07
	NON	Carré □	0.77 ± 0.01	0.88 ± 0.02	0.74 ± 0.00	0.16 ± 0.01
		Croix +	0.76 ± 0.01	0.98 ± 0.01	0.74 ± 0.01	0.21 ± 0.08
		Copyright ©	0.76 ± 0.00	0.84 ± 0.02	0.73 ± 0.01	0.27 ± 0.12

TABLE 1 : Défense après l'entraînement pour le cas IID et non-IID.

		Notre défense		RLR		fCD		Med	
Dataset	IID	PB	EA	PB	EA	PB	EA	PB	EA
MNIST	Oui	0.94	0.11	0.97	0.01	0.90	0.18	0.94	0.12
	Non	0.88	0.15	0.62	0.13	0.73	0.34	0.84	0.19
FashionMNIST	Oui	0.81	0.15	0.84	0.06	0.72	0.41	0.82	0.14
	Non	0.74	0.16	0.52	0.16	0.60	0.35	0.68	0.26

TABLE 2 : Comparaison à l'état-de-l'art pour les cas IID et non-IID pour l'attaque sur motif carré.

Pour adresser le challenge de l'hétérogénéité des données entre les clients (non-IID), nous forçons les distributions des classes des clients à être différentes, en n'allouant que 5 classes sur les différentes 10 classes à chaque client.

Pour la reconstruction du déclencheur, nous fixons $\gamma = 0.1$ et $n = 50$ (voir partie 3.1). Enfin, nous répétons chaque expérience cinq fois.

4.2 Résultats

Nos résultats sont reportés au sein des Tables 1 et 2. Tab 1 résume les performances (PB : précision sur données benignes et EA : efficacité de l'attaque) obtenues par le modèle fédéré sans et avec notre défense. En l'absence de défense, il apparaît qu'il est très facile d'empoisonner le modèle avec un motif élémentaire qui sera fortement corrélé à la classe cible. De plus, cet empoisonnement se fait sans impacter PB ce qui confirme que les attaques par porte dérobée sont difficilement détectables. Enfin, de façon logique l'hétérogénéité des données va entraîner une diminution de la précision du modèle fédéré par rapport au cas IID (sans toutefois impacter l'efficacité de l'attaque qui est indépendante de la distribution des données). En présence de notre défense, nous constatons une baisse significative de l'efficacité de l'attaque (nous sommes souvent proche de 0.1 qui correspond à une défense parfaite dans le cas 10 classes) sans dégrader de façon significative les performances du modèle sur la tache principale (en moyenne, nous observons une dégradation de 2%). Notre défense apparaît cependant moins efficace contre l'attaque à motif copyright, parce que ce motif est plus difficile à estimer et la reconstruction n'est pas si fiable. Mais même dans ce cas complexe, l'utilisation de notre défense est avantageux car elle diminue de façon significaive l'impact d'une BA.

Dans Tab 2, nous comparons notre défense aux approches état-de-l'art qui ne reposent pas sur des informations supplémentaires transmises par des clients. A l'exception de fCD dont l'ajout de bruit va dégrader les performances du modèle, l'ensemble des défenses fournit des résultats similaires pour le cas IID. Cependant, pour le cas non-IID, notre défense permet d'obtenir le meilleur compromis pour diminuer l'attaque de façon significative sans dégrader la tâche principale (surtout sur FashionMNIST). Par manque de place, nous n'avons pu reporter les résultats obtenus avec 40 et 60% de clients malveillants mais il s'avère que les performances de notre défense ne sont pas impactées par ce paramètre contrairement aux autres approches proposées dans la littérature.

5 Conclusion

Dans ce papier, nous avons proposé une nouvelle défense contre les BA basée sur deux algorithmes : la reconstruction du motif déclencheur par inversion de modèle et l'élagage d'un réseau de neurones. Notre défense est robuste au cas non-IID inhérent au processus d'AF et ne s'appuie sur aucune information supplémentaire envoyée par les clients autre que celle nécessaire à l'AF. Les expériences réalisées sur MNIST et FashionMNIST montre l'intérêt d'utiliser notre défense par rapport aux approches état-de-l'art.

Références

[1] Peva BLANCHARD *et al.* : Machine learning with adversaries : Byzantine tolerant gradient descent. *Advances in Neural Information Processing Systems*, 30, 2017.

[2] Matt FREDRIKSON, Somesh JHA et Thomas RISTENPART : Model inversion attacks that exploit confidence information and basic countermeasures. *In Proc. of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015.

[3] Tianyu GU, Kang LIU, Brendan DOLAN-GAVITT *et al.* : Badnets : Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[4] Peter KAIROUZ *et al.* : Advances and open problems in federated learning. *arXiv :1912.04977*, 2019.

[5] Brendan MCMAHAN *et al.* : Communication-efficient learning of deep networks from decentralized data. 2016.

[6] Mustafa Safa OZDAYI *et al.* : Defending against backdoors in federated learning with robust learning rate. *In Proc. of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9268–9276, 2021.

[7] Ziteng SUN, Peter KAIROUZ *et al.* : Can you really backdoor federated learning ? *arXiv :1911.07963*, 2019.

[8] Chen WU *et al.* : Mitigating backdoor attacks in federated learning. *arXiv preprint arXiv :2011.01767*, 2020.

[9] Dong YIN, Yudong CHEN *et al.* : Byzantine-robust distributed learning : Towards optimal statistical rates. *In ICML*, pages 5650–5659, 2018.

[10] Chen ZHAO *et al.* : Federatedreverse : A detection and defense method against backdoor attacks in federated learning. *In Proc. of the ACM Workshop on Information Hiding and Multimedia Security*, pages 51–62, 2021.