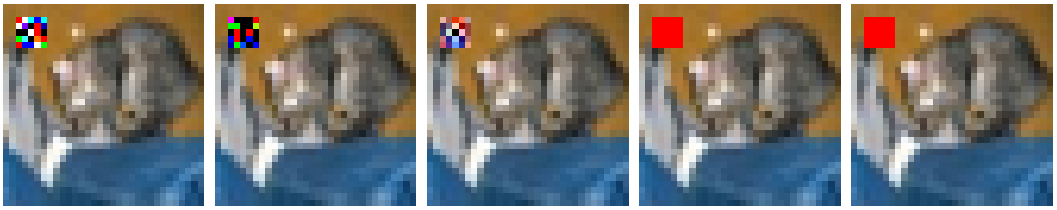


(a) FLTrojan (b) F3BA (c) CerP (d) Neurotoxin (e) DBA

Figure 7: Trigger patterns of evaluated attacks on FedAvg, with P = 2 compromised clients.

A Additional Experiment Details

A.1 Experimental Setup in Figure 1

The preliminary experiment in Figure 1 has the same experimental setup as described in Section 4.1. In particular, We use FedAvg [2] as the server-side aggregation rule. We set the number of compromised clients $P = 1$ in the preliminary experiment. We denote the attack success rate on the global model as global ASR. We further denote the ASR on the local model after local training as the local ASR. When the compromised client is selected by the server, we calculate and update the local ASR after the compromised client optimizes the backdoor trigger and trains its local model on the poisoned local training dataset.

A.2 Details of Attacks

A3FL: A3FL formulates the trigger optimization as a bi-level optimization problem. A3FL jointly optimizes the adversarial model $f_{\theta'_t}$ with the trigger pattern $\mathbf{\Delta}$. A3FL optimizes the adversarial model using SGD with a learning rate of 0.01, a momentum of 0.9, and a weight decay of 0.0005. A3FL updates the trigger pattern using PGD with a step size of 0.01. The trigger optimization is repeated for 40 epochs. We show the trigger pattern of A3FL in Figure 7a.

F3BA [10]: F3BA directly manipulates a part of local model weights to inject the backdoor via sign flipping. F3BA further jointly optimizes the trigger pattern and the local model weights to maximize the difference between latent representations of clean and backdoored samples, thus achieving higher attack performance. The trigger of F3BA is a squared patch. We show the trigger pattern of F3BA in Figure 7b.

CerP [9]: CerP jointly optimizes the trigger pattern and the local model weights to improve the backdoor effectiveness. Furthermore, CerP aims to improve the backdoor stealthiness by adopting L2-norm regularization to limit the difference between local model weights and global model weights. Therefore CerP can tune the local model to fit the backdoor-poisoned data without inducing large biases in the local model weights. The trigger of CerP is shown in Figure 7c.

Neurotoxin [12]: Neurotoxin only updates unimportant model weights to avoid conflicts with other clean clients. The importance of model weights is determined by the magnitude of their gradients. Model weights with a higher gradient in previous rounds are considered to be more important (frequently updated by other clients). Following the settings in [12], we only update the last 95% important model weights. Neurotoxin uses a fixed trigger pattern, as shown in Figure 7d.

DBA [11]: DBA is a distributed backdoor attack designed to utilize the distributed nature of FL. DBA splits the trigger into different clients. Each client uses a different trigger to attack the FL system during the training stage. In the inference stage, the attacker uses the joint trigger to activate the injected backdoor. The trigger in [11] was designed as several parallel white lines placed at the upper left corner of the input images. This trigger design is not compatible with our attack setting and we can hardly control the attack budget introduced by the trigger following [11]. Therefore in our implementation, we also use a squared patch as the trigger for DBA, as shown in Figure 7e. We randomly split the squared patch into four sub-triggers and these sub-triggers are iteratively used during the attack.

A.3 FL defenses

Norm Clipping (NC) [17]: NC clips clients' updates that are larger than a pre-defined threshold. NC can effectively limit clients' behavior to prevent the global model from being overwhelmed by a few clients. By default, we set the threshold to 1.

(weak) Differential Privacy (DP) [17]: DP adds Gaussian noise $\boldsymbol{z} \sim \mathcal{N}(0, \sigma^2 I)$ to clients' updates to perturb carefully crafted malicious updates. Note that this defense is not designed for privacy, so the Gaussian noise is relatively smaller than that adopted in differential privacy. By default, we set $\sigma = 0.002$.

Robust Learning Rate (RLR) [18]: RLR aims to maximize the agreement on updating direction across clients to mitigate potential attacks. It is inspired by that the behavior of a compromised client is commonly different from other benign clients. For instance, a compromised client may want to enlarge some model parameters while most benign clients are trying to reduce them. When clients disagree on the updating direction of a parameter, RLR flips the learning rate on the parameter to maximize the loss instead.

CRFL [19]: CRFL adopts three techniques to mitigate backdoor attacks on FL. CRFL first clips clients' updates as Norm Clipping does. In our experiments, we set the clipping threshold as 1. CRFL then adds Gaussian noise $\boldsymbol{z} \sim \mathcal{N}(0, \sigma^2 I)$ to clients' updates as DP does. In our experiments, we set $\boldsymbol{\delta} = 0.002$ and we discuss the impact of σ on CRFL in Appendix B.5. Finally, CRFL creates several perturbed models by adding independently sampled Gaussian noise to the global model and adopts majority voting for prediction. In our experiments, CRFL creates 5 different perturbed models for prediction at each FL communication round.

Median [22]: Median uses the coordinate-wise median value of updates from all clients to update the global model. Median can effectively exclude clients that upload overwhelming updates. However, the Median tends to heavily degrade the model utility.

Deepsight [20]: Deepsight adopts three different distance matrices to measure the distances between each client. Deepsight then clusters clients according to different distance matrices and only accepts clients that are in the same cluster across different matrices. The first distance matrix is smaller when the updates in the last layer from clients are similar. The second distance matrix is the L2 distance between the last layer's weight across each client. The third distance matrix is the L2 distance between the outputs of two local models given a batch of randomly generated input images. Deepsight adopts DBSCAN [40] to cluster selected clients. Finally, clusters including potentially malicious clients that have a larger distance from other clusters will be excluded. In our experiments, we set the batch size of randomly generated inputs to 256.

Bulyan [23]: Bulyan first excludes potentially malicious clients from all selected clients and then uses the coordinate-wise median value of updates from remaining clients to update the global model. In the first step, $2f$ clients with the highest pairwise Euclidean distances are excluded. In the second step, Bulyan picks $M - 4f$ clients from the remaining $M - 2f$ clients that are closest to the median by coordinate. In our experiments, we set $f = 2$.

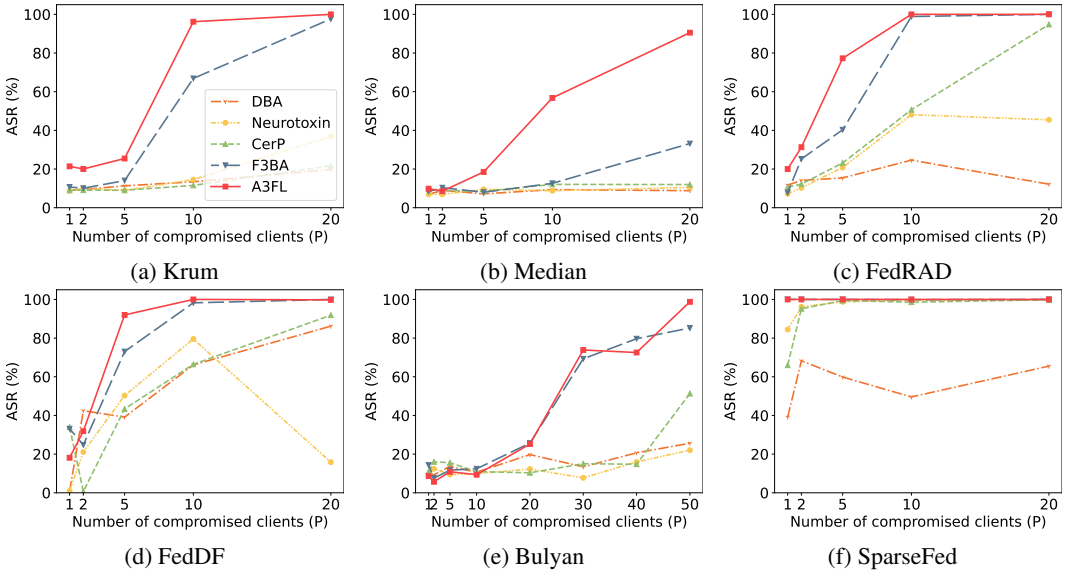
FedDF [24]: FedDF uses the mean output of all client models as the supervisory signal to distill the next round global model. In particular, FedDF firstly aggregates all selected clients (the same as FedAvg) to obtain a teacher model. Then the server trains the global model to minimize the Kullback Leibler divergence between the logits of the global and teacher model on a set of unlabeled inputs. In our experiments, the learning rate for updating the global model is 0.002 and we train the global model for one epoch at each FL communication round.

FedRAD [25]: FedRAD is an extension of FedDF, which assigns a weight to each client model based on their median scores. These scores indicate the frequency with which the prediction of the client model becomes the median value of predictions from all client models. FedRAD then utilizes weighted model aggregation to produce the next round global model. In our experiments, we also update the global model with a learning rate of 0.002 for one epoch at each FL communication round.

Krum [21]: Krum selects clients that have the smallest L2 distances to other clients. Only the clients selected by Krum will be used to update the global model. Since Kurm drops most updates from clients, it can achieve strong robustness. However, Krum also affects the accuracy of the model.

Table 2: A3FL maintains the utility of global models on TinyImageNet.

Defense	FedAvg	NC	RLR	Median	DSight	Bulyan	Krum	SFed	CRFL	DP	FedDF	FedRAD
ACC(%)	55.45	55.31	55.34	17.12	53.71	11.19	42.87	57.39	53.58	53.38	25.31	23.12
BAC(%)	55.25	54.98	55.28	20.92	53.44	7.33	42.35	57.08	53.45	53.17	24.90	22.57



(a) Krum (b) Median (c) FedRAD

(d) FedDF (e) Bulyan (f) SparseFed

Figure 8: Comparing performances of different attacks on CIFAR-10.

SparseFed [26]: SparseFed is proposed to mitigate model poisoning attacks in FL. SparseFed aggregates client updates normally but only updates the top-k highest magnitude elements. It is inspired by that attackers commonly move in distinct directions from the majority of clean clients. Therefore the top-k highest magnitude elements involve less poisoned updates from attackers. In our experiments, we update the top-95% highest magnitude elements.

B Additional Experimental Results

B.1 A3FL maintains the model utility

We show the accuracy of the global model on TinyImagenet when the attacker presents (BAD) or not (ACC) in Table 2. In particular, we record the accuracy on clean tasks when no attackers are involved to obtain the accuracy (ACC). We further record the accuracy on clean tasks when there are 20 compromised clients among all clients to obtain the backdoor accuracy (BAC). We set the number of compromised clients P to 20 since more compromised clients are likely to result in a higher decrease in clean accuracy. Therefore if A3FL can maintain the model utility even with 20 compromised clients, we can conclude that A3FL is highly stealthy. Note that we use the mean value of ACC and BAC in the attack window (between the 1,900th communication round and the 2,000th communication round) to verify the utility of global models since the server continuously updates the global model. Therefore, using the mean accuracy as the measurement standard can accurately reflect the impact of attacks on the model utility, and eliminate randomness.

As shown in Table 2, the accuracy of the global model does not degrade much when attackers are presented. This indicates that A3FL preserves the accuracy of global models so it is stealthy enough to not be discovered. The differences between ACCs and BACs are within 0.5% in most cases. The highest drop in clean accuracy is observed when the defense mechanism is Bulyan. However, Bulyan significantly degrades the model's accuracy to only 11.19%. The low accuracy indicates that the model is highly random, so even though A3FL causes the model's accuracy to drop to 7.33%, we cannot solely conclude that A3FL will reduce the model utility. In general, A3FL does not influence the global model utility. We also observe a similar phenomenon on CIFAR-10, as shown in Table 1.

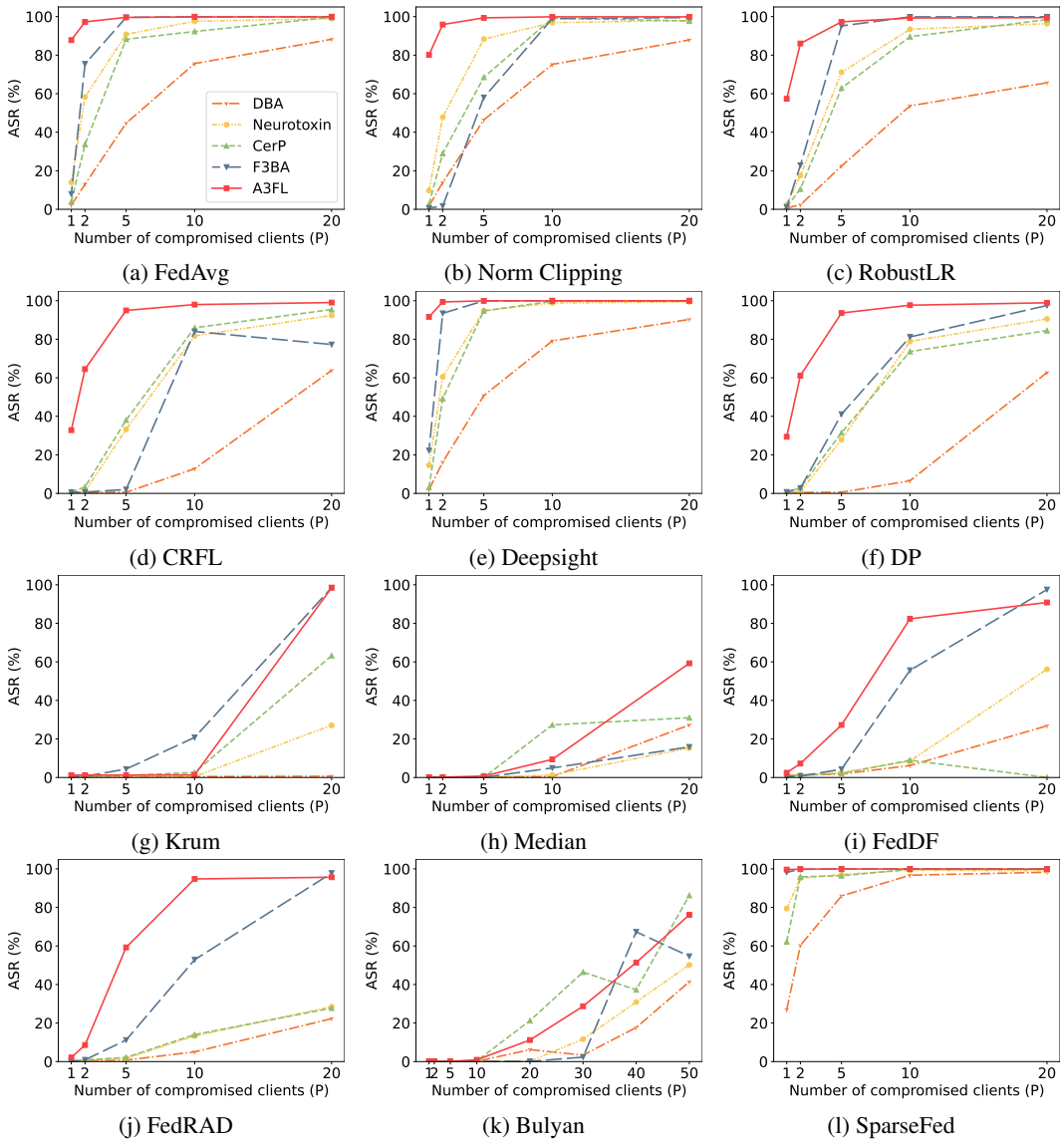


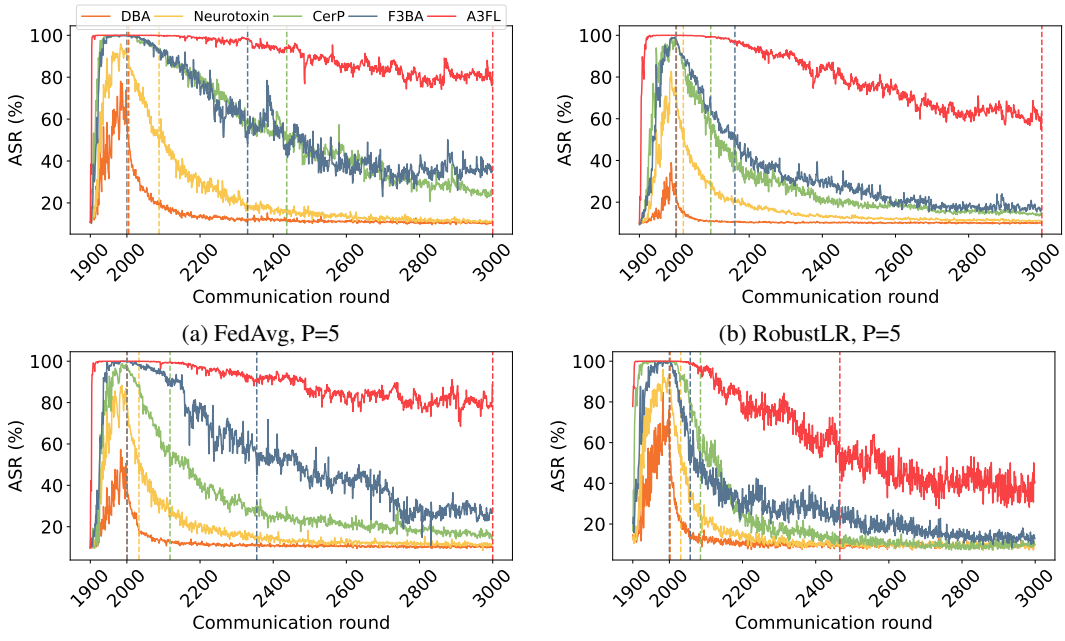
Figure 9: Comparing performances of different attacks on TinyImageNet.

B.2 A3FL achieves higher ASRs

We compare the performance of attacks on CIFAR-10 against defenses that are not designed for backdoor attacks in Figure 8. Observe that A3FL achieves the highest ASR under most settings. When the defense is Median, A3FL is the only attack that can achieve high ASR (over 80%). We further show the attacker performance of A3FL on TinyImagenet in Figure 9 and we can observe a similar phenomenon.

B.3 A3FL has a longer lifespan

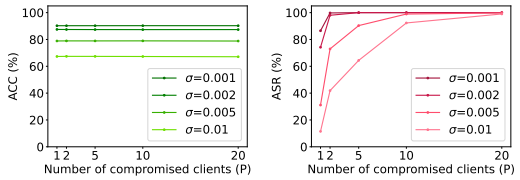
In Figure 10, we show that A3FL has a significantly longer lifespan than other baselines with different defenses applied. For instance, when the defense is RobustLR, A3FL can still achieve an ASR of 62.37% at 1000 rounds after the attack ends. In contrast, the attack success rates of other attacks drop below 50% in less than 150 rounds. Note that when we use CRFL, we set the number of compromised clients $P = 20$ since when there are only 5 compromised clients, all attacks except A3FL failed to achieve high ASR (see Figure 2).



(a) FedAvg, P=5 (b) RobustLR, P=5

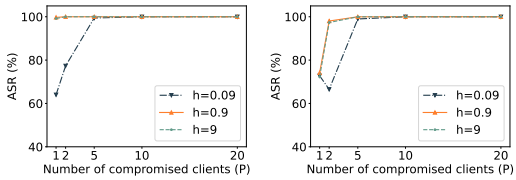
(c) Deepsight, P=5 (d) CRFL, P=20

Figure 10: A3FL has a longer lifespan.



(a) ACC (b) ASR

Figure 11: Attack performances against CRFL with different σ.



(a) Norm Clipping (b) CRFL

Figure 12: Attack performances under different Dirichlet concentration parameters.

B.4 Ablation study on component importance

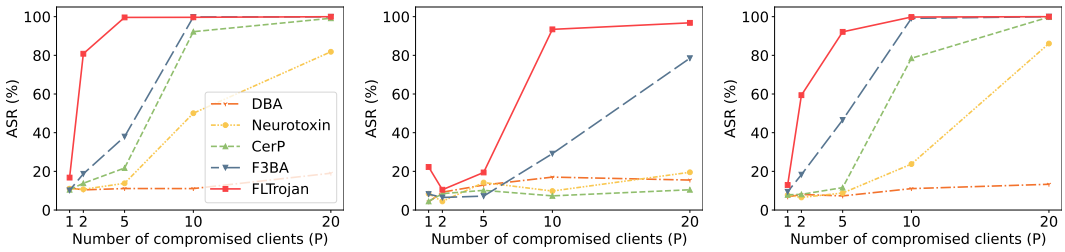
We study the effectiveness of A3FL with or without the adversarial adaptation loss to test the effectiveness of components under FedAvg with $P = 20$ compromised clients among all clients. As shown in Table 3, the adversarial adaptation loss can effectively improve the durability of A3FL. Observe that A3FL can achieve an ASR of 97.66% at 500 communication rounds after the attack and 86.65% at 1,000 communication rounds after the attack. In comparison, A3FL without the adversarial adaptation loss exhibits ASRs that are 4.31% and 15.64% lower than A3FL at these two points.

Table 3: Effect of different components in A3FL.

ASR(%) ↓ Rounds after attack →	0	500	1000
A3FL without adversarial adaptation	100.0	93.35	69.01
A3FL	100.0	97.66	84.65

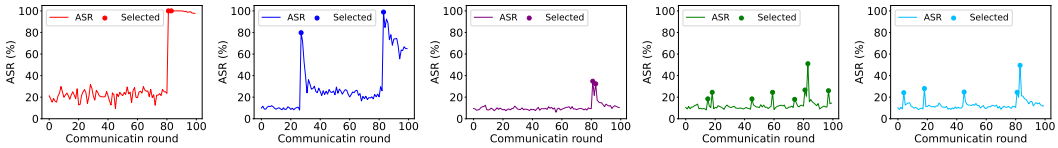
B.5 Impact of σ on CRFL Effectiveness

Figure 11 shows the ACC and ASR when applying CRFL with different σ. Observe that as the σ increases, CRFL can achieve better robustness, indicated by lower ASR. However, the ACC of the global model also drops from 90.25% to 67.33% rapidly, as σ increases from 0.001 to 0.01, which



(a) Norm Clipping (b) Krum (c) CRFL

Figure 13: Attack performances when the attack starts at the first communication round.



(a) A3FL (b) F3BA (c) CerP (d) DBA (e) Neurotoxin

Figure 14: ASRs against Krum.

is unacceptable. Furthermore, when there are more compromised clients, A3FL can still achieve high ASR even with a large $\sigma = 0.01$. We can thus conclude that CRFL can not sufficiently mitigate A3FL with different σ.

B.6 Impact of Data Heterogeneity

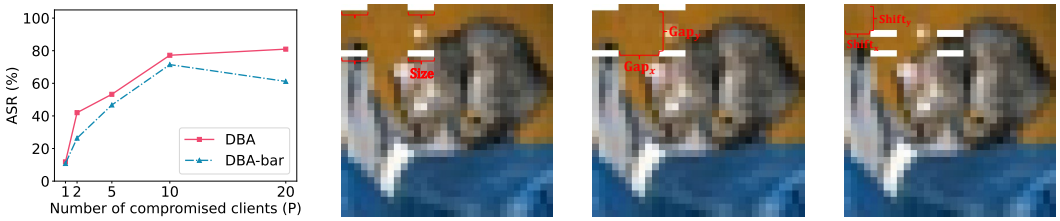
We adjust the Dirichlet concentration parameter $h = 0.09, 0.9, 9$ to study whether data heterogeneity influences the performance of A3FL. As shown in Figure 12, A3FL can achieve high ASR regardless of different h. When the defense is Norm Clipping and $h = 0.09$, A3FL achieves lower ASR. This can be explained by that a smaller h indicates a more non-i.i.d data distribution. Therefore, the local training set held by the attacker is far from the global data distribution, which increases the difficulty of injecting the backdoor. However, the attack success rate is still high (over 60%) and quickly increases as the number of compromised clients increases.

B.7 The impact of attack window

We evaluate A3FL against baseline attacks when the attack window starts at the first communication round and ends at the 100th communication round. As shown in Figure 13, A3FL can still remarkably outperform other baseline attacks. For instance, when the defense mechanism is Norm Clipping and there are 5 compromised clients, the gaps of ASR between A3FL and other baseline attacks are at least 62.4%, which is even larger than the gap under default settings. However, we also observe that when the attack starts from the first communication round and there are only a few compromised clients (1 or 2), ASRs of all attacks decrease in comparison to ASRs under default settings. This can be explained by that at the beginning of the training process, the global model changes a lot so the backdoor is easily erased when there are only a few compromised clients.

B.8 Case study on Krum

We perform a case study on Krum to gain insight into why A3FL outperforms other baselines. In Figure 14 we record the ASRs and put a "·" notation on the line if Krum selects an attacker-compromised client at that round. Recall that Krum selects one client at each round and only uses the selected client updates to update the global model. Therefore, the chance that a compromised client is selected by the server increases if the backdoor is more stealthy. We have the following observations: 1) fixed-trigger attacks are more frequently selected by the server, while trigger-optimization attacks are selected twice only; 2) fixed-trigger attacks achieve lower ASR even if selected by the server. However, observe that once selected, A3FL quickly achieve 100% ASR, which is because A3FL can maintain higher ASR when transferred to the global model as stated above. A3FL is also durable after



(a) ASR (b) Trigger Size (c) Trigger Gap (d) Trigger Location

Figure 15: Attack performances of DBA using original trigger design. (a) DBA-bar denotes DBA attack with the original trigger design proposed in [11], in which the trigger consists of four white bars. While DBA denotes the DBA attack with the trigger designed as a red square. (b) Trigger size refers to the length of each white bar. (c) Trigger gap $\{\text{Gap}_x, \text{Gap}_y\}$ refers to the distance between each bar. (d) Trigger location $\{\text{Shift}_x, \text{Shift}_y\}$ represents the distance from the trigger to the edge of the image.

being selected, leading to a higher ASR at the end of the attack. In comparison, F3BA is selected on the 26th round and achieves $\approx 80\%$ ASR. But the ASR quickly drops after that. CerP is also selected twice, but it cannot achieve as high ASR as A3FL and F3BA do, which is caused by the strict regularization on the local model bias. In addition, the ASR of CerP also drops quickly when the compromised clients are not selected by the server.

B.9 The impact of DBA trigger pattern

In our experiments, we set the trigger pattern of DBA to be a red square at the upper left corner. However, in [11], the trigger is designed as four white lines. We, therefore, discuss the performance of DBA when using the original trigger design. The original trigger design of DBA is determined by three hyperparameters: trigger size (TS), trigger gap (TG), and trigger location (TL). In particular, the trigger gap consists of a horizontal gap (Gap_x) and a vertical gap (Gap_y). The trigger location consists of a horizontal shift (Shift_x) and a vertical shift (Shift_y). We explain these hyperparameters in Figure 15b,15c, and 15d respectively. Following the default settings in [11], we set $\{\text{TS}, \text{TG}, \text{TL}\} = \{4, (6, 6), (0, 0)\}$.

We compare the attack performance of DBA and DBA-bar (DBA with original trigger design) in Figure 15a. Observe that with the original trigger design, DBA-bar achieves an even lower ASR. This phenomenon supports that the default trigger design in our experiments does not degrade the attack performance of DBA. In contrast, DBA can even achieve a higher ASR without the original trigger design.