ĐẠI HỌC QUỐC GIA TP. HCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO TỔNG KẾT
ĐỀ TÀI KHOA HỌC VÀ CÔNG NGHỆ SINH VIÊN NĂM 2021

Tên đề tài tiếng Việt:

TẤN CÔNG ĐẦU ĐỘC CHỐNG LẠI MÔ HÌNH HỌC MÁY FEDERATED LEARNING

Tên đề tài tiếng Anh:

AN EMPIRICAL STUDY ON POISONING ATTACKS AGAINST FEDERATED LEARNING SYSTEM

Khoa/ Bộ môn: Mạng máy tính và Truyền thông

Thời gian thực hiện: 06 tháng

Cán bộ hướng dẫn: ThS. Nguyễn Thanh Hoà

Tham gia thực hiện

TT	Họ và tên	Chịu trách nhiệm	Điện thoại	Email
1.	**Trần Nhật Tân**	Chủ nhiệm	0902506931	17521019@gm.uit.edu.vn
2.	Phạm Ngọc Tâm	Tham gia	0795541213	18521371@gm.uit.edu.vn
3.	Đoàn Thanh Phương	Tham gia	0944022034	18521267@gm.uit.edu.vn

Mẫu F6

ĐẠI HỌC QUỐC GIA TP. HCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Ngày nhận hồ sơ	
Mã số đề tài	
(Do CQ quản lý ghi)	

BÁO CÁO TỔNG KẾT

Tên đề tài tiếng Việt:

TẤN CÔNG ĐẦU ĐỘC CHỐNG LẠI MÔ HÌNH HỌC MÁY FEDERATED LEARNING

Tên đề tài tiếng Anh:

AN EMPIRICAL STUDY ON POISONING ATTACKS AGAINST FEDERATED LEARNING SYSTEMS

Ngày ... tháng ..... năm	*Ngày ... tháng ..... năm ....*
Cán bộ hướng dẫn	Sinh viên chủ nhiệm đề tài
(Họ tên và chữ ký)	*(Họ tên và chữ ký)*
Nguyễn Thanh Hoà	**Trần Nhật Tân**

Mẫu F6

THÔNG TIN KẾT QUẢ NGHIÊN CỨU

1. Thông tin chung

- Tên đề tài:

TẤN CÔNG ĐẦU ĐỘC CHỐNG LẠI MÔ HÌNH HỌC MÁY FEDERATED LEARNING

- Chủ nhiệm: **Trần Nhật Tân**

- Thành viên tham gia 1: **Phạm Ngọc Tâm**

- Thành viên tham gia 2: **Đoàn Thanh Phương**

- Cơ quan chủ trì: Trường Đại học Công nghệ Thông tin.

- Thời gian thực hiện: 06 tháng.

2. Mục tiêu

Những tiến bộ vượt bậc của công nghệ, sự phát triển không ngừng về khả năng tính toán, số lượng thiết bị và sự bùng nổ về dữ liệu lớn đã tạo các điều kiện thuận lợi thúc đẩy sự phát triển mạnh mẽ của các kỹ thuật Machine Learning (ML – Học máy), đặc biệt là Deep Learning (DL – Học sâu), góp phần giải quyết nhiều vấn đề quan trọng của đời sống như y tế, môi trường và đô thị thông minh. Với các mô hình *ML* phân tán, dữ liệu cần được thu thập từ nhiều nguồn như các thiết bị *IoT* (Internet of Things), điện thoại di động và lưu trữ - xử lí tập trung để cung cấp cho ứng dụng *ML* trung tâm. Tuy nhiên, hướng tiếp cận phổ biến này ngày càng đối mặt với nhiều thách thức, không đảm bảo cho sự phát triển bền vững như: 1) việc thu thập, lưu trữ dữ liệu tập trung đòi hỏi chi phí lớn và tốn kém nhiều thời gian; 2) mối quan ngại về tính riêng tư của dữ liệu nhạy cảm của người dùng ngày càng được đặc biệt quan tâm [1]. Do đó, tình trạng đói dữ liệu (data hungriness) vẫn đang là một thách thức nghiêm trọng đối với các ứng dụng ML truyền thống. Để giải quyết hiệu quả vấn đề trên, mô hình học máy để đảm báo tính riêng tư của dữ liệu – Federated Learning (FL – Học hợp tác) lần đầu tiên được giới thiệu bởi nhóm nghiên cứu từ Google [2] vào năm 2016. Với *FL*, việc huấn luyện mô hình *FL* sẽ diễn ra trên chính thiết bị của người dùng và cùng kết hợp các đánh giá để huấn luyện mô hình toàn cục (global model).

Dữ liệu riêng tư của người dùng được lưu trữ hoàn toàn trên thiết bị của họ và không chia sẻ hay gửi đến máy chủ tập trung hoặc người dùng khác. Thay vào đó, chỉ có những cập nhật của mô hình được gửi đến máy chủ tập trung để tổng hợp và xây dựng mô hình toàn cục.

Sự quan tâm của các nhà nghiên cứu đối với Federated Learning đang ngày càng gia tăng kể từ ngày đầu tiên công bố. Ví dụ tiêu biểu cho mô hình này là ứng dụng bàn phím ảo Gboard của Google[3]. Những bài toán *ML* liên quan đến tính riêng tư của dữ liệu tồn tại trước đây có thể được giải quyết bằng cách áp dụng mô hình *FL* (ví dụ các ứng dụng y tế trên thiết bị thông minh cần truy cập dữ liệu nhạy cảm như thông tin sức khỏe, ứng dụng xe tự hành, trợ lí ảo). Nhiều tổ chức, nhóm nghiên cứu đã tích cực phát triển những nền tảng để hỗ trợ cho Federated Learning như Tensorflow Federated của Google, PySyft từ Openmined; FATE từ Webank hay Owkin hiện đang nghiên cứu phát triển nền tảng *FL* ứng dụng trong lĩnh vực y học [3]. Mặc dù mô hình *FL* có thể mang lại những cải tiến đáng kể về đảm bảo tính riêng tư của dữ liệu, nhưng nhiều nghiên cứu vẫn chứng minh được rằng cách thiết kế của các mô hình *FL* tồn tại nhiều rủi ro bảo mật có thể bị khai thác bởi kẻ tấn công ở cả trong và ngoài hệ thống [1]. Ví dụ, *FL* vẫn đối mặt với các phương pháp tấn công gây ảnh hưởng đến hiệu suất và độ chính xác của mô hình thông qua việc tận dụng cơ chế hoạt động của bộ tổng hợp trung tâm (aggregator). Cụ thể, các phương pháp tấn công đầu độc dữ liệu và mô hình (data and model poisoning attacks) [4]–[8] gây ảnh hưởng nghiêm trọng đến độ chính xác của mô hình; hay thậm chí với phương pháp tấn công tinh vi, khó phát hiện như tấn công backdoor [9], [10] không làm ảnh hưởng đến độ chính xác của mô hình mà hiệu quả tấn công đầu độc vẫn đạt được rất cao. FL vẫn đang là hướng nghiên cứu rất mới và đang được đặc biệt quan tâm bởi cộng đồng nghiên cứu. Để tiếp cận và có đánh giá thực tế về *FL*, trong nghiên cứu khoa học này, nhóm tác giả sẽ có phân tích cụ thể hơn về tính bảo mật của mô hình Federated Learning cũng như khảo sát các mối đe dọa tiềm năng. Từ đó, tác giả sẽ tiến hành thực nghiệm các phương pháp tấn công đầu độc chống lại mô hình *FL*, đánh giá và đưa ra nhận định về các rủi ro khi ứng dụng mô hình *FL* vào thực tế cũng như bước đầu đề xuất các giải pháp phòng thủ và hạn chế ảnh hưởng của các tấn công đầu độc cho mô hình *FL* trong tương lai.

3. Tính mới và sáng tạo

Federated Learning vẫn đang là hướng nghiên cứu rất mới và đang được đặc biệt quan tâm bởi cộng đồng nghiên cứu trên khắp thế giới. Và đang được áp dụng để giải quyết các bài toán liên quan đến quyền riêng tư của dữ liệu người dùng và khối lượng dữ liệu để huấn luyện một mô hình học máy. Tuy nhiên, khi áp dụng mô hình FL vào thực tế, tồn tại những rủi ro đe dọa đến sự an toàn của hệ thống và dữ liệu của người dung. Cụ thể, những phương pháp tấn công đã được nghiên cứu để chống lại mô hình FL hoạt động trong thực tế bằng những cách đơn giản nhất.

4. Tóm tắt kết quả nghiên cứu

Trong đề tài nghiên cứu này, nhóm tác giả đã xây dựng thành công mô hình mô phỏng Federated Learning và thực hiện 2 phương pháp tấn công đầu độc dữ liệu chống lại mô hình FL với nhiều kịch bản tấn công như sau:

1. Phương pháp A: *Đầu độc mô hình Federated Learning sử dụng dữ liệu có sẵn.*
2. Phương pháp B: *Cải tiến phương pháp tấn công đầu độc sử dụng Mạng đối nghịch tạo sinh (GAN)*

Qua các kịch bản thực nghiệm, nhóm tác giả đã chứng minh được rằng tấn công đầu độc dữ liệu bằng phương pháp thay đổi nhãn dữ liệu trên mô hình FL là có hiệu quả và khả thi trong thực tế. Kẻ tấn công có thể gây ảnh hưởng tiêu cực đến mô hình toàn cục làm cho mô hình hoạt động không như mong muốn tại các vòng bị tấn công. Thực nghiệm còn cho thấy với tỉ lệ kẻ tấn công tăng dần thì hiệu quả của phương pháp tấn công tốt hơn.

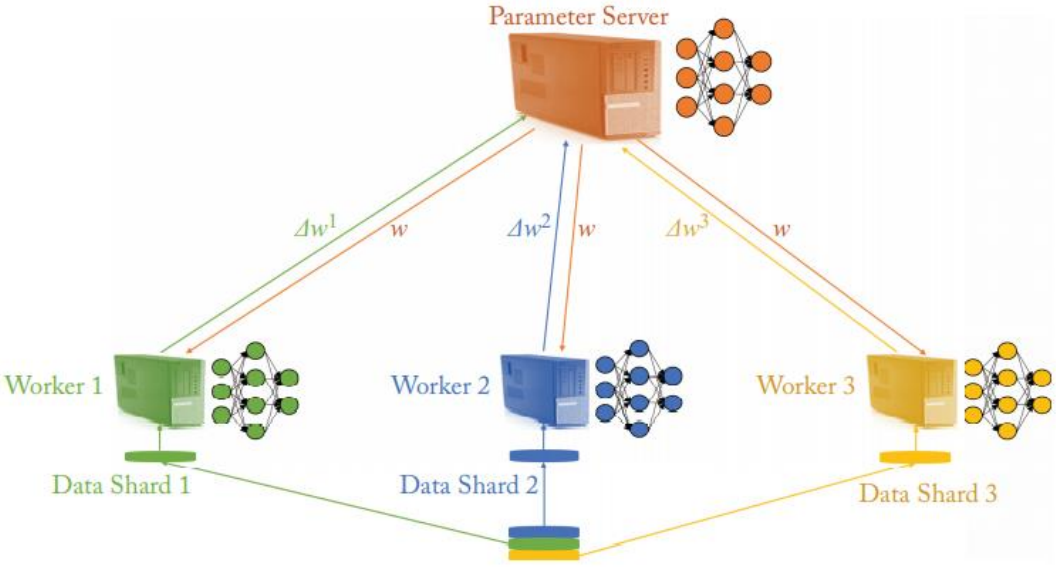
4.1. Nghiên cứu cơ sở lý thuyết và các nghiên cứu liên quan

4.1.1. Mô hình học máy phân tán

Nhu cầu về trí tuệ nhân tạo đã tăng lên đáng kể trong thập kỉ qua đồng thời mang lại nhiều thách thức về khả năng tính toán và tính riêng tư của dữ liệu. Mặt khác, con người luôn muốn nâng cao hiệu quả cũng như độ chính xác của học máy trong các ứng đụng thực tế, điều này đòi hỏi cần huấn luyện mô hình với một lượng dữ liệu lớn và đa dạng. Hơn nữa, các mô hình học máy quy mô nhỏ có thể được đào tạo với lượng dữ liệu vừa đủ nhưng đối với các mô hình lớn hơn như mạng học sâu (mạng nơ-ron)

sẽ luôn phát triển theo cấp số nhân với số lượng tham số do đó nhu cầu xử lý dữ liệu huấn luyện đã vượt mức sự gia tăng khả năng tính toán của máy tính. Vì vậy, học máy phân tán đã ra đời như một giải pháp hiệu quả bằng cách phân phối khối lượng công việc học tập trên nhiều máy và biến học máy tập trung thành một hệ thống phân tán, từ đó khối lượng công việc lớn sẽ được phân chia thực hiện trên nhiều máy một cách dễ dàng. Hơn thế nữa, học máy phân tán còn là một giải pháp tăng cường tính riêng tư của dữ liệu người dùng trong học máy hiện nay [11].

Ý tưởng của học máy phân tán (distributed learning) là sử dụng nhiều nút cùng đảm nhiệm việc thực hiện các thuật toán *ML* hoặc *DL* để cải thiện hiệu suất, đảm bảo tính riêng tư, tăng lượng dữ liệu huấn luyện và có thể phát triển những mô hình phức tạp hơn. Ví dụ, *Hình 4.1* mô tả một hệ thống học phân tán bao gồm ba nút tính toán và một máy chủ tổng hợp, để giảm bớt tính toán cho các nút, dữ liệu huấn luyện được chia thành ba phần cho ba nút và các nút này thực hiện huấn luyện tại cục bộ. Sau đó các nút này gửi các kết quả cập nhật mô hình về phía máy chủ tổng hợp nơi các cập nhật từ các nút được tính trung bình và cho ra cập nhật cuối cùng. Với giải pháp này, học máy phân tán có thể phân chia công việc huấn luyện cho các nút và đảm bảo riêng tư của dữ liệu.



Hình 4.1. Mô hình học máy phân tán [12]

Từ định nghĩa trên, học máy phân tán có thể được phân thành hai loại: scalability-motivated (thúc đẩy khả năng mở rộng) và privacy-motivated (đảm bảo quyền riêng tư). *Scalability-motivated* là mô hình được thiết kế để giải quyết nhu cầu mở rộng và tập trung tối ưu khả năng tính toán của các nút trong hệ thống học máy. Ngược lại,

Scalability-motivated tập trung giải quyết các nhu cầu về đảm bảo tính riêng tư của người dùng bởi vì tính riêng tư dữ liệu của người dùng hiện đang là mối quan tâm toàn cầu.

4.1.2. Mô hình học máy hợp tác

4.1.2.1. Giới thiệu

Mô hình học máy hợp tác (Federated Learning - *FL*) lần đầu tiên được giới thiệu bởi nhóm nghiên cứu từ Google vào năm 2016 nhằm giải quyết những vấn đề mà mô hình học máy phân tán đang đối mặt. Federated Learning là cách thiết lập mô hình học máy mà trong đó nhiều thiết bị (clients) cùng hợp tác để giải quyết vấn đề liên quan đến học máy, dưới sự điều phối của một máy chủ trung tâm hoặc nhà cung cấp dịch vụ.

FL là một cách tiếp cận học máy phân tán trong đó nhiều người dùng cùng huấn luyện một mô hình chung trong khi vẫn đảm bảo quyền riêng tư cho dữ liệu của người dùng vì không phải chuyển đến một máy chủ hoặc trung tâm dữ liệu trung gian nào khác [13], [14]. Ban đầu, *FL* được đề xuất để xử lý dữ liệu *IID* của các đặc trưng giống nhau trong thiết bị di động - trong *FL* gọi là *Cross-Device*. Sau đó, khái niệm *FL* được mở rộng sang dữ liệu phân tán của các công ty, tổ chức hay nhiều vùng địa lý khác nhau - trong *FL* gọi là *Cross-Silo*. Tóm lại, hệ thống *FL* là một hệ thống phân tán để quản lý quá trình huấn luyện ở nhiều nơi khác nhau với các tài nguyên phân tán [14] mà vẫn đảm bảo được tính riêng tư của dữ liệu.

Mô hình *FL* giải quyết những vấn đề liên quan đến quyền riêng tư của dữ liệu và giúp giảm chi phí trong quá trình huấn luyện mô hình học máy. Trong quá trình tham gia vào hệ thống *FL*, dữ liệu của mỗi nút tham gia được lưu trữ cục bộ và sẽ không chia sẻ hay chuyển đi với bất kì thực thể nào. Với cách tiếp cận đó, mô hình *FL* đã giải quyết những vấn đề liên quan đến quyền riêng tư của dữ liệu. Sau khi hoàn thành quá trình huấn luyện mô hình cục bộ, các nút sẽ thực hiện gửi những cập nhật của mô hình cục bộ cho máy chủ trung tâm hoặc nhà cung cấp dịch vụ. Việc chỉ gửi những cập nhật của mô hình thay vì khối lượng lớn dữ liệu gốc đã giúp giảm chi phí truyền thông giữa các nút và máy chủ trung tâm một cách đáng kể.

4.1.2.2. Phân loại

Cross-devices: là mô hình FL với lượng thiết bị tham gia rất lớn có thể là các thiết bị di động hoặc thiết bị *Internet of Things* có thể lên đến hàng tỉ thiết bị. Tuy nhiên trong mô hình này các thiết bị đạt độ tin cậy thấp, do chúng không dùng bất kỳ thông tin gì để định danh và số lượng thiết rất lớn, đa dạng. Các thiết bị trong quá trình huấn luyện có thể gặp các vấn đề về tiêu thụ điện năng hay về kết nối mạng dẫn đến mất kết nối trong quá trình huấn luyện. Cross-device được Google sử dụng trong ứng dụng bàn phím Gboard [13], và một số tính năng khác trong điện thoại Pixel và trong Android Messages [13].

Cross-silo: là mô hình *FL* trong đó có số lượng nút ít hơn Cross-devices, với số lượng khoảng từ 2 đến 100 thiết bị tham gia đến từ nhiều tổ chức thuộc nhiều lĩnh vực khác nhau (như y tế và tài chính) hoặc từ các trung tâm dữ liệu ở các vị trí khác nhau cùng chia sẻ dữ liệu để huấn luyện một mô hình nhưng không thể chia sẻ các dữ liệu đó một cách trực tiếp. Cross-silo đạt độ tin cậy cao, do chúng đều được định danh rõ ràng. Các thiết bị hầu như luôn khả dụng nên việc huấn luyện mô hình hầu như ít gặp sự cố [13].

4.1.2.3. Vòng đời của mô hình học hợp tác

Vòng đời của mô hình FL có thể được tóm tắt thành 4 giai đoạn sau [13]:

- **Giai đoạn khởi tạo**: khởi tạo một mô hình *FL*, được sử dụng để giải quyết một vấn đề học máy cụ thể, ví dụ như bài toán phân loại với một số yêu cầu nhất định, ví dụ: yêu cầu về độ chính xác. Sau đó, mô hình học máy được thiết lập cho phù hợp với các kịch bản *FL*. Ví dụ: nếu dữ liệu được phân phối có nhiều loại khác nhau, thì mô hình học máy sẽ được phân vùng để xử lý dữ liệu phân tán.
- **Giai đoạn huấn luyện**: trong giai đoạn này, một chiến lược huấn luyện, bao gồm các quá trình huấn luyện song song và tổng hợp, được sử dụng để cập nhật các tham số mô hình và thậm chí cả cấu trúc của mạng giao tiếp giữa các nút, để cải thiện độ chính xác và khả năng tổng quát hóa của mô hình *FL*.
- **Giai đoạn đánh giá**: phân tích hiệu suất của các mô hình *FL* đã được đào tạo sau khi được áp dụng. Sau đó, các mô hình *FL* có hiệu suất tốt nhất sẽ

được chọn. Nếu các mô hình *FL* không đáp ứng các yêu cầu, mô hình *FL* nên được sửa đổi, hoặc giai đoạn huấn luyện nên được thực hiện lại

- **Giai đoạn triển khai**: đưa mô hình *FL* được chọn sau khi đánh giá vào một kịch bản thực tế để xử lý dữ liệu, nếu như mô hình đạt được kết quả tối ưu và bảo mật hơn giải pháp sử dụng học máy với dữ liệu tập trung, đồng thời giải quyết được vấn đề bài toán (phân loại) đặt ra.

4.1.2.4. Quá trình huấn luyện

Máy chủ trung tâm điều phối quá trình huấn luyện, bằng cách lặp lại những bước dưới đây cho đến khi quá trình kết thúc [13]:

- **Lựa chọn người tham gia**: Máy chủ trung tâm sẽ lựa chọn tập hợp các nút đáp ứng đủ các điều kiện để tham gia vào quá trình *FL*. Ví dụ: các thiết bị điện thoại di động chỉ có thể tham gia vào khi đang sạc pin, có kết nối wifi,…
- **Quảng bá mô hình toàn cục**: Những nút được lựa chọn ở **bước 1** sẽ thực hiện tải xuống các trọng số của mô hình toàn cục hiện tại và chương trình huấn luyện từ máy chủ trung tâm.
- **Huấn luyện ở các nút**: Mỗi thiết bị được lựa chọn sẽ thực hiện huấn luyện mô hình và tính toán cập nhật của mô hình.
- **Máy chủ tổng hợp kết quả từ các nút**: Máy chủ trung tâm thu thập cập nhật của những nút được chọn, sau đó sử dụng thuật toán tổng hợp để cải thiện mô hình toàn cục.
- **Máy chủ cập nhật mô hình toàn cục**: Máy chủ trung tâm cập nhật mô hình toàn cục dựa trên tính toán tổng hợp ở **bước 4**. Trở lại **bước 1**.

4.1.2.5. Các mối đe dọa và rủi ro

Mặc dù mô hình *FL* có thể giải quyết được tính riêng tư dữ liệu của người dùng so với phương pháp học tập trung truyền thống nhưng *FL* vẫn thừa hưởng các mối đe dọa và rủi ro của các cuộc tấn công trong học máy. Nhóm tác giả đưa ra một số phương pháp tấn công phổ biến trong mô hình *FL* như sau [15]:

- **Poisoning attack**: Tấn công đầu độc là cuộc tấn phổ biến và nghiêm trọng bởi vì mô hình toàn cục có thể bị nhiễm độc từ nhiều nguồn như các kết quả cập nhật của mô hình cục bộ, máy chủ độc hại và nhiều nguồn khác.
- **Inference attack**: Tương tự như mối đe dọa đầu độc, các mối đe dọa của tấn công suy luận cũng có rất nhiều nguồn gây nên lỗ hổng.
- **Backdoor attack**: Mối đe dọa của các cuộc tấn công cửa hậu rất khó phát hiện và có thể vượt mặt các cơ chế xác thực của mô hình toàn cục.
- **GAN**: Mạng đối nghịch tạo sinh (GAN) cũng là một trong những lí do khiến cho *FL* mắc lỗ hỗng do không thể đoán trước và có khả năng ảnh hưởng đến bảo mật và quyền riêng tư của dữ liệu người dùng.
- **Malicious server**: Máy chủ độc hại thừa hưởng những lỗ hổng mở của các máy chủ vật lý hoặc điện toán đám mây, do đó trong *FL* cũng có những lỗ hổng này.
- **Communication bottlenecks**: Mỗi lần cập nhật mô hình toàn cục, các nút phải gửi các kết quả huấn luyện từ mô hình cục bộ về máy chủ sau nhiều lần lặp có thể gây ra tắc nghẽn trong quá trình giao tiếp. Do đó ảnh hưởng đến thời gian huấn luyện và độ ổn định của hệ thống.

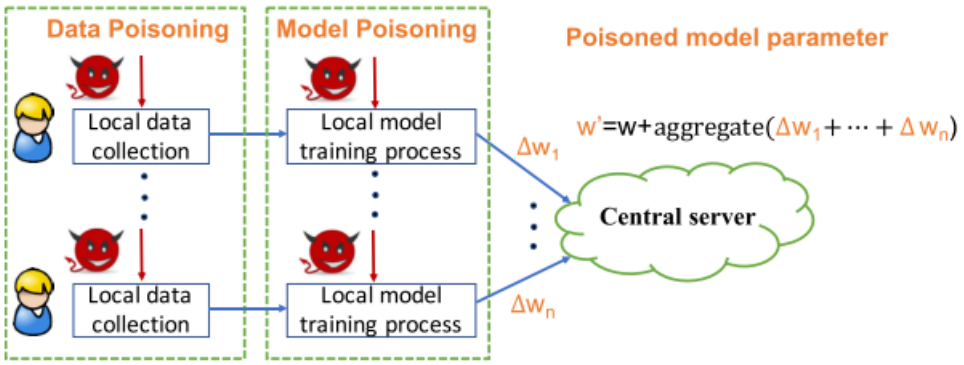
4.1.3. Tấn công đầu độc chống lại mô hình học máy hợp tác

Trong tấn công đầu độc mô hình học *FL*, người tấn công có thể là một trong những người tham gia huấn luyện của mô hình *FL*, nên có thể biết được và sửa đổi các cài đặt thông số mô hình. Do đó, có thể gọi là một cuộc tấn công *white-box* [13] để phân loại các cuộc tấn công có thể dựa vào mục tiêu của người tấn công và có thể chia làm 2 loại:

- **Tấn công không có mục tiêu**: là kiểu tấn công chỉ nhằm mục đích giảm độ chính xác của mô hình học *FL*.
- **Tấn công có mục tiêu**: ngược lại, tấn công có mục tiêu làm cho cho mô hình học *FL* cho ra các kết quả đúng theo ý muốn của người tấn công.

Tấn công đầu độc trong suốt quá trình huấn luyện có thể được thực hiện trên dữ liệu hoặc trên mô hình. *Hình 4.2* cho thấy các kết quả cập nhật mô hình có thể bắt nguồn từ hai loại tấn công đầu độc: (1) tấn công đầu độc dữ liệu trong suốt quá trình thu thập dữ liệu; và (2) tấn công đầu độc mô hình trong suốt quá trình huấn luận. Nhìn

chung, cả hai loại tấn công này đều nhằm cố gắng thay đổi cách hoạt động của mô hình mục tiêu theo nhiều cách khác nhau. Nếu người tấn công có thể chiếm quyền điều khiển máy chủ FL, chúng có thể dễ dàng thực hiện cả hai loại tấn công có mục tiêu và không có mục tiêu trên mô hình đã được huấn luyện.



Hình 4.2. Tấn công đầu độc dữ liệu và mô hình trong FL

4.1.3.1. Tấn công đầu độc dữ liệu

Trong tấn công đầu độc dữ liệu (data poisoning attacks) cũng chia làm hai loại: **(1)** Giữ nguyên nhãn [16] và **(2)** Thay đổi nhãn (label-flipping) [17].

Tấn công giữ nguyên nhãn giả sử rằng người tấn công không thể thay đổi nhãn của bất kì dữ liệu huấn luyện nào bởi vì có một quá trình kiểm tra xem dữ liệu phải được gắn nhãn một cách chính xác. Ngược lại, trong tấn công đầu độc *label-flipping*, người tấn công có thể đưa vào một số mẫu dữ liệu và chúng mong muốn các mẫu dữ liệu này sẽ được phân loại sang một nhãn khác thuộc tập huấn luyện mà chúng mong muốn.

Một ví dụ thường thấy của *label-flipping* là phương pháp tấn công thay đổi nhãn dữ liệu (label-flipping attacks) [4], [18]. Các nhãn của tập dữ liệu huấn luyện ban đầu của một lớp được thay thế bởi một nhãn của lớp khác trong khi các đặc trưng của dữ liệu vẫn được giữ nguyên không thay đổi. Một ví dụ cụ thể hơn, những người tấn công trong hệ thống có thể đầu độc dữ liệu của chúng bằng cách thay đổi tất cả các nhãn có giá trị 1 thành giá trị là 7. Nếu cuộc tấn công thành công, mô hình mục tiêu sẽ không thể phân loại đúng các dữ liệu thuộc nhãn giá trị 1 và dự đoán sai ra nhãn có giá trị 7 [1]. Một cuộc tấn công khác không dễ dàng để triển khai tuy nhiên lại phù hợp với ngữ cảnh thực tế là tấn công *backdoor*[17]. Ở đây, một người tấn công có thể chỉnh sửa các đặc trưng hoặc một vùng nhỏ của dữ liệu huấn luyện gốc để có thể gắn

một *backdoor* vào mô hình, do đó mô hình có thể thay đổi cách hoạt động tùy vào mục đích của người tấn công nếu dữ liệu đầu vào chứa các đặc trưng độc hại.

Tấn công đầu độc dữ liệu trong mô hình học *FL* có thể được thực hiện bởi bất kì nút tham gia nào. Mức độ ảnh hưởng đến mô hình *FL* tùy thuộc vào số lượng nút tham gia trong hệ thống thực hiện tấn công và lượng dữ liệu huấn luyện bị đầu độc

4.1.3.2. Tấn công đầu độc mô hình

Tấn công đầu độc mô hình nhằm mục đích là để đầu độc các kết quả của mô hình cục bộ trước khi chúng được gửi lên máy chủ hoặc chèn các *backdoor* ẩn vào mô hình toàn cục [10].

Trong tấn công đầu độc có mục tiêu, mục đích của người tấn công là làm cho mô hình FL phân loại sai một tập các dữ liệu đầu vào với độ sai lệch cao. Tất nhiên, các dữ liệu đầu vào này không được chỉnh sửa để gây ra dự đoán sai trong quá trình thực nghiệm đánh giá. Hơn nữa, việc dự đoán sai là kết quả của việc tính toán nhầm lẫn trong quá trình huấn luyện. Cụ thể, việc đầu độc dữ liệu trên các kết quả cập nhật mà trong đó một tập con kết quả bị đầu độc được gửi đến máy chủ tại bất kì thời điểm nào. Những kết quả bị đầu độc này có thể được tạo ra bằng cách chèn *backdoor* ẩn, và thậm chí một cuộc tấn công *single-shot* có thể đủ thêm *backdoor* vào mô hình [17].

Tấn công đầu độc mô hình hiệu quả hơn nhiều so với tấn công đầu độc dữ liệu trong ngữ cảnh *FL* bằng cách phân tích một cuộc tấn công đầu độc mô hình có mục tiêu, trong đó một người tham gia độc hại đơn lẻ, không thông đồng với bất kì người tham gia nào khác nhằm mục đích khiến mô hình bỏ sót phân loại một tập hợp các đầu vào đã chọn với độ lệch [19].

Thực tế, tấn công đầu độc mô hình cũng có thể xảy ra trong cuộc tấn công đầu độc dữ liệu trong *FL*, bởi vì tấn công đầu độc dữ liệu cũng có thể làm thay đổi kết quả cập nhật mô hình được gửi lên phía máy chủ *FL* tại một thời điểm. Tấn công đầu độc mô hình yêu cầu khả năng kỹ thuật phức tạp và cần tài nguyên tính toán lớn [18].

4.1.4. Mô hình Mạng đối nghịch tạo sinh (GAN)

Mạng đối nghịch tạo sinh (Generative Adversarial Network - *GAN*) được giới thiệu bởi Goodfellow và cộng sự vào năm 2014 [20] đã dẫn đến sự cải thiện đáng kể trong quá trình tạo ra hình ảnh, dự đoán video và một số lĩnh vực khác. Ý tưởng của *GAN* là huấn luyện đồng thời một bộ phân biệt (Discriminator) và một bộ sinh dữ liệu

(Generator). *Discriminator* được huấn luyện để có khả năng phân biệt các mẫu dữ liệu thật với các mẫu dữ liệu giả do bộ sinh dữ liệu tạo ra. *Generator* sử dụng đầu vào từ các dữ liệu ngẫu nhiên (random noise) và được huấn luyện để tạo ra các mẫu dữ liệu giả mà bộ phân biệt không phân biệt được với các mẫu dữ liệu thật.

ACGAN là một mô hình GAN tạo ra dữ liệu có điều kiện và hoạt động tương tự như *CGAN* [21]. Đầu vào của bộ phân biệt là dữ liệu thật hoặc dữ liệu giả được tạo ra từ bô sinh dữ liệu và nhãn của dữ liệu đó. Khác với *CGAN*, kết quả của bộ phân biệt bao gồm xác suất phân loại thật giả và kết quả phân loại nhãn của dữ liệu đầu vào.

Trong đề tài Nghiên cứu khoa học này, nhóm tác giả sẽ sử dụng mô hình *ACGAN* để sinh ra dữ liệu để tấn công đầu độc mô hình *FL*. Lí do chính tác giả sử dụng *ACGAN* là vì cấu trúc của mô hình *ACGAN* phù hợp với môi trường *FL*, khi mà có thể sử dụng *Discriminator* để làm mô hình toàn cục chung. Kẻ tấn công có thể dễ dàng xây dựng và huấn luyện mô hình *ACGAN* cục bộ để sinh dữ liệu tấn công đầu độc.

4.2. Phân tích thiết kế mô hình

4.2.1. Đầu độc mô hình Federated Learning từ dữ liệu tấn công có sẵn

Từ những phân tích và đánh giá ở trên cho thấy rằng nhân tố chính của *label-flipping attacks* (tấn công đầu độc bằng cách thay đổi nhãn dữ liệu) là dữ liệu mà *attacker* có để thực hiện huấn luyện *local model*. Để đánh giá tổng quan phương pháp tấn công đầu độc này, nhóm tác giả bước đầu xây dựng mô hình tấn công dựa trên giả thiết *attacker* ban đầu đã chiếm được quyền kiểm soát của một người tham gia bình thường hoặc *attacker* đã có sẵn dữ liệu huấn luyện khi tham gia vào quá trình *FL*. Nhóm tác giả đã tiến hành phân tích và đánh giá phương pháp *label-flipping attacks* được đề xuất bởi *Vale Tolpegin và các cộng sự* [4]. Sau đó, thực hiện cải tiến xây dựng mô hình tấn công dựa vào đề xuất trước đó để có sự nhìn nhận và đánh giá khách quan hơn về phương pháp *label-flipping attacks*.

4.2.2. Cải tiến phương pháp tấn công đầu độc sử dụng mô hình Mạng đối nghịch tạo sinh (GAN)

Từ các phân tích và giả thiết đối với phương pháp tấn công thay đổi nhãn dữ liệu huấn luyện sử dụng dữ liệu có sẵn, vấn đề chính để thực hiện phương pháp *label-flipping attacks* là *attacker* phải có được dữ liệu bao gồm một số lớp nhãn. Tuy nhiên,

trong mô hình FL, dữ liệu của người tham gia được huấn luyện cục bộ và *attacker* không biết được những dữ liệu này. Do đó việc triển khai tấn công đầu độc dữ liệu trở nên khó khăn hơn trong trường hợp kẻ tấn công không sở hữu đầy đủ lớp nhãn dữ liệu huấn luyện hoặc có ít số lượng mẫu dữ liệu để thực hiện tấn công. Để giải quyết những vấn đề trên, tác giả xây dựng mô hình Mạng đối nghịch tạo sinh (GAN), cụ thể là Auxiliary Classifier GAN (ACGAN) trong môi trường *FL*. Với những đặc điểm của mô hình *ACGAN*, nhóm tác giả nhận thấy với cấu trúc mô hình *ACGAN* thì có thể sẽ triển khai thực nghiệm được trong môi trường *FL* mà vẫn đảm bảo được chức năng của *global model*.

5. Tên sản phẩm: ACGP – Đầu độc mô hình Federated Learning

6. Hiệu quả, phương thức chuyển giao kết quả nghiên cứu và khả năng áp dụng

6.1. Thông tin phần cứng

Tác giả xây dựng môi trường thực nghiệm cho cả hai phương pháp tấn công đầu độc dữ liệu được thực nghiệm trong Khóa luận này, được mô tả chi tiết như sau:

Phương pháp tấn công đầu độc sử dụng dữ liệu có sẵn: Ở phương pháp tấn công này, nhóm tác giả xây dựng mô hình mô phỏng *FL* gần giống với lý thuyết, được triển khai cục bộ giữa các máy tính cùng mạng với nhau. Cụ thể chi tiết cấu hình như sau:

- Máy ảo 1 (Server): Ubuntu Server 18.04 - 8GB RAM - 8 CPUS.
- Máy ảo 2 (Client): Ubuntu Server 18.04 - 32GB RAM - 24 CPUS.

Phương pháp tấn công đầu độc sử dụng mô hình GAN: Nhóm tác giả xây dựng mô hình mô phỏng FL một cách đơn giản nhất dựa vào thư viện PyTorch. Thực nghiệm chỉ tiến hành trên một máy duy nhất, có cấu hình như sau:

- Ubuntu Server 18.04 - 32GB RAM - 16 CPUS.

6.2. Tập dữ liệu thực nghiệm

Để đánh giá 2 phương pháp tấn công được đề xuất trong Nghiên cứu khoa học này, nhóm tác giả tiến hành các nhiệm vụ phân loại khác nhau dựa trên 2 tập dữ liệu khác nhau là MNIST, Fashion-MNIST. Thông tin về các tập dữ liệu được mô tả tại *Bảng 6.1*.

Datasets	Labels	Input	Training samples	Testing samples

MNIST	10	28x28x1	60.000	10.000
F-MNIST	10	28x28x1	50.000	10.000

Bảng 6.1. Thông tin các tập dữ liệu thực nghiệm

MNIST [1]: Tập dữ liệu này có 10 lớp nhãn bao gồm các chữ số viết tay từ 0 đến 9, trong đó có 70.000 hình ảnh được chia thành tập dữ liệu huấn luyện (60.000 hình ảnh) và tập dữ liệu kiểm tra (10.000 hình ảnh). Các ảnh đầu vào được chuẩn hoá bằng cách chuyển thành ảnh mức xám (grayscale) và có kích thước mỗi ảnh là 28x28 điểm ảnh (pixel).

Fashion-MNIST [2](F-MNIST): Tập dữ liệu này giống như MNIST, có 10 lớp nhãn bao gồm ảnh của các loại áo quần như *coat, shirt, dress*,… Số lượng mẫu dữ liệu là 60.000 được chia thành 2 tập dữ liệu, 50.000 mẫu cho tập dữ liệu huấn luyện và số còn lại được chia cho tập dữ liệu kiểm tra. Các ảnh đầu vào được chuẩn hoá bằng cách chuyển thành ảnh *grayscale* và có kích thước mỗi ảnh là 28x28 pixels.

6.3. Thực nghiệm và đánh giá kết quả

6.3.1. Tấn công đầu độc sử dụng dữ liệu có sẵn

6.3.1.1. Cấu hình thực nghiệm

Thiết lập mô hình (Model settings): Tác giả sử dụng mô hình Convolution neutral network (CNN) để xây dựng mô hình phân loại ảnh cho cả 2 tập dữ liệu *MNIST* và *F-MNIST*.

Cấu hình quá trình huấn luyện (Training configurations): Tác giả tiến hành lựa chọn và cấu hình các thông số liên quan đến việc tấn công và quá trình huấn luyện mô hình. Cụ thể như sau:

- **Tổng số người tham gia** (Participants): Tổng số người tham gia vào quá trình FL trong phương pháp tấn công này sẽ được chọn là 50.
- **Số người tham gia trong một vòng** (Participants per round): Số người tham gia trong một vòng được lựa chọn từ *Participants*, mặc định là 5.

[1] **https://www.kaggle.com/c/digit-recognizer/data**

[2] **https://www.kaggle.com/zalando-research/fashionmnist**

- **Dữ liệu của người tham gia**: Dữ liệu huấn luyện của người tham gia vào quá trình FL sẽ bằng số lượng mẫu dữ liệu của từng tập dữ liệu *MNIST* và *F-MNIST* chia đều cho số người tham gia, cụ thể là 1200 mẫu dữ liệu cho mỗi người tham gia.
- *Thông số của quá trình huấn luyện*: *Attacker* sẽ thực hiện huấn luyện *local model* với các thông số giống với người tham gia bình thường.

Cấu hình tấn công (Attack configurations): Nhóm tác giả tiến hành thực nghiệm tấn công đầu độc với số lượng kẻ tấn công tăng dần. Với các cách thiết lập thay đổi nhãn của từng tập dữ liệu được tác giả mô tả và giải thích cụ thể ở phần sau.

6.3.1.2. Thực nghiệm A1: Đánh giá ảnh hưởng của việc chọn cặp nhãn dữ liệu tấn công

Để đánh giá ảnh hưởng của việc chọn cặp nhãn dữ liệu để thực hiện tấn công đầu độc, tác giả tiến hành thực nghiệm với mỗi tập dữ liệu trên 3 cặp thiết lập với tỉ lệ nhầm lẫn tăng dần (khi không có tấn công đầu độc) ở 3 mức độ gồm: thấp nhất (L), trung bình (M), và cao (H). Cùng với đó, nhóm tác giả đánh giá độ chính xác của *global model* khi xảy ra tấn công trong nhiều vòng với từng thiết lập của mỗi tập dữ liệu.

Dựa trên giá trị *baseline misclassification* (m_cnt) của mỗi thiết lập khi không có tấn công đầu độc, tác giả lựa chọn các cặp nhãn dữ liệu tương ứng với từng mức độ dựa vào nghiên cứu trong [4] như sau:

- **F-MNIST**: (1) 1 → 3(L, m_cnt = 17); (2) 4 → 6 (M, m_cnt = 75); (3) 6 → 0 (H, m_cnt = 137).
- **MNIST**: (1) 3 → 9 (L, m_cnt = 3); (2) 6 → 0 (M, m_cnt = 8); (3) 7 → 2 (H, m_cnt = 17).

Sau đó, nhóm tác giả tiến hành tấn công với tỉ lệ kẻ tấn công (m%) tăng đần từ 2% đến 50% trên tổng số người tham gia vào quá trình *FL* để đánh giá kết quả và mức độ ảnh hưởng của từng mốc tỉ lệ. Độ chính xác của *global model* được theo dõi qua từng vòng cho đến khi kết thúc quá trình *FL*. Để tính sự chênh lệch của *source class recall* của từng thiết lập đối với *baseline misclassification*, nhóm tác giả sử dụng *Công thức (1)*:

$$\Delta source\_class\_recall = source\_class\_recall_{NP} - source\_class\_recall_{m}(\%)$$

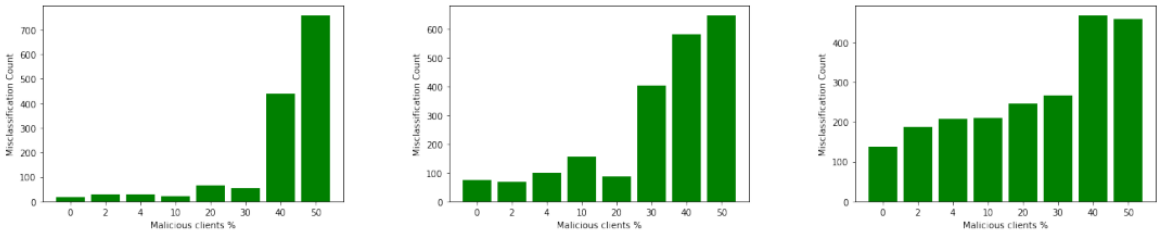
(1)

$c_{src} \rightarrow c_{target}$	$m\_cnt_{targe}^{src}$	Tỉ lệ kẻ tấn công tham gia (m%)						
		2	4	10	20	30	40	50
Fashion-MNIST								
1 → 3	17	0.96	1.16	0.6	5.16	3.64	42.34	74.44
4 → 6	75	-1.59	2.09	10.48	2.07	35.47	56.12	62.96
6 → 0	137	11.11	12.69	10.99	17.62	18.3	45.44	45.36
MNIST								
3 → 9	3	-1.23	-0.59	2.99	32.5	18.97	43.68	60.75
6 → 0	8	0.67	0.42	0.96	3.61	27.85	54.55	76.74
7 → 2	17	-0.6	0.72	15.97	31.73	12.9	31.8	62.96

Bảng 6.2. Sự chênh lệch về tỉ lệ của source class recall với các thiết lập source → target đối với tỉ lệ *baseline misclassification* khi không có tấn công của 2 tập dữ liệu *F-MNIST* và *MNIST*.

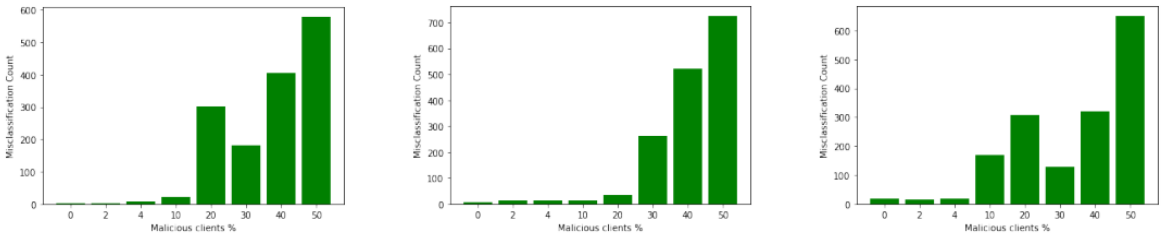
Bảng 6.2 mô tả kết quả về sự chênh lệch tỉ lệ của *source class recall* với các thiết lập khác nhau cho mỗi tập dữ liệu. Xem xét kết quả đối với mỗi tập dữ liệu như sau:

- **Đối với tập dữ liệu F-MNIST**, khi tỉ lệ *attacker* thấp (m < 20%), tấn công trên cặp dữ liệu H (6 → 0) sẽ đạt hiệu quả tốt nhất, cụ thể là tăng 17.62% khi m = 20%. Tuy nhiên, khi tăng tỉ lệ *attacker* từ 30% đến 50%, tấn công trên các cặp dữ liệu M và L đạt hiệu quả cao hơn, cụ thể khi tỉ lệ *attacker* m = 50% thì tỉ lệ chênh lệch này đạt 74.44% với thiết lập L, trong khi đó ở thiết lập H chỉ đạt 45.36%.
- **Đối với tập dữ liệu MNIST**, các xu hướng về sự chênh lệch source class recall trên tập dữ liệu *MNIST* tương tư so với tập dữ liệu *F-MNIST*. Đối với m = 10% thì tỉ lệ chênh lệch của mức H (7 → 2) là 15.97%. Tuy nhiên đối với m > 20% thì các mức thiết lập L và M lại đạt được hiệu quả cao hơn khi tỉ lệ chênh lệch lớn hơn so với thiết lập H, cụ thể ở mức m = 50% thì tỉ lệ chênh lệch ở thiết lập M là 76.74%.



(a) $1 \rightarrow 3$ (b) $4 \rightarrow 6$ (c) $6 \rightarrow 0$

Hình 6.1. Số lượng phân loại sai từ c_src thành c_target cho từng thiết lập trong tập dữ liệu Fashion-MNIST.



(a) $3 \rightarrow 9$ (b) $6 \rightarrow 0$ (c) $7 \rightarrow 2$

Hình 6.2. Số lượng phân loại sai từ c_src thành c_target cho từng thiết lập trong tập dữ liệu MNIST.

Để giải thích về tỉ lệ chênh lệch giữa các mức L, M và H, tác giả dựa vào số lượng phân loại sai cơ bản của từng cặp $c_{src} \rightarrow c\_target$ tức là số lượng mẫu dữ liệu bị phân loại sai từ $c_{src} \rightarrow c\_target$ khi không có tấn công đầu độc. Khi tỉ lệ *attacker* m% tham gia vào quá trình *FL* đạt một mức độ nhất định thì số lượng này sẽ có sự chênh lệch giữa các cặp nhãn đã được chọn ở trên. Hình 6.1 và 6.2 biểu thị số lượng phân loại sai từ c_{src} thành c_target cho từng thiết lập $c_{src} \rightarrow c\_target$ trên 2 tập dữ liệu *F-MNIST* và *MNIST*. Kết quả cho thấy, khi tỉ lệ kẻ tấn công đạt từ 40% đến 50% thì số lượng phân loại sai tăng đáng kể. Cụ thể như sau:

- **Đối với tập dữ liệu *F-MNIST***, số lượng phân loại sai của thiết lập $6 \rightarrow 0$ khi không có tấn công là 137 trong khi với 2 thiết lập còn lại $1 \rightarrow 3$ và $4 \rightarrow 6$ lần lượt là 17 và 75. Đối với trường hợp m < 20% thì không có sự chênh lệch đáng kể ở các thiết lập L và M, với m = 30% thì thiết lập L vẫn không cho thấy sự chênh lệch rõ ràng khi với M. Với m lần lượt là 30% và 40% thì M cho thấy tỉ lệ chênh lệch tăng nhanh hơn so với các thiết lập L và H. Khi tỉ lệ *attacker* đạt bằng nửa số lượng người tham gia m = 50%, tỉ lệ chênh lệch ở thiết lập L cao hơn rất nhiều so với các thiết lập còn lại, cụ thể là 743 so với 572 và 322 của mức M và H.

- **Đối với tập dữ liệu *MNIST***, vì độ chính xác của mô hình khi không có tấn công rất cao và số lượng phân loại sai cơ bản thấp. Cụ thể số lượng phân loại sai cơ bản của từng thiết lập $c_{src} \rightarrow c\_target$ là 17, 8, 3 cho lần lượt 7 → 2, 6 → 0 và 3 → 9. Thực nghiệm cho thấy với tỉ lệ kẻ tấn công cao hơn 20% thì số lượng phân loại sai đã có sự thay đổi rõ rệt. Cụ thể, ở trường hợp tỉ lệ kẻ tấn công đạt mức độ cao nhất, số lượng phân loại sai là 651, 726, 579 và số lượng chênh lệch là 634, 716, 573 cho lần lượt từng thiết lập 7 → 2, 6 → 0 và 3 → 9.

Kết quả thực nghiệm cho thấy với cặp nhãn có số lượng phân loại sai cơ bản cao thì số lượng phân loại sai ở một khoảng tỉ lệ phần trăm kẻ tấn công nhất định sẽ thấp hơn so với một lớp nhãn có tỉ lệ phân loại sai cơ bản thấp. Dựa vào đó, tác giả cũng có thể nhận định rằng khi thực hiện *label-flipping attacks* trên mô hình *FL*, việc chọn cặp dữ liệu $c_{src} \rightarrow c\_target$ để tấn công thực sự không cần thiết và không phải là yếu tố quyết định ảnh hưởng đến hiệu quả của cuộc tấn công. Tuy nhiên, việc này sẽ làm ảnh hưởng đến độ chính xác của *global model* qua các vòng khi thực hiện tấn công đầu độc, tác giả sẽ trình bày chi tiết thực nghiệm này ở phần tiếp theo.

6.3.1.3. Thực nghiệm A2: Đánh giá ảnh hưởng của các nhãn dữ liệu không bị tấn công đến độ chính xác của mô hình toàn cục

Để đảm bảo độ chính xác của mô hình toàn cục chỉ bị ảnh hưởng do tác động duy nhất của c_{src}^{recall} và chứng minh phương pháp tấn công không ảnh hưởng đến các nhãn dữ liệu khác (ngoài c_{src} và c_{target}), nhóm tác giả tiến hành ghi lại kết quả của từng lớp nhãn dữ liệu để thực hiện phân tích và đánh giá. Bên cạnh đó, nhóm tác giả muốn chứng minh một đặc điểm của phương *label-flipping attacks* là *tấn công có mục tiêu* (targeted attack).

Nhóm tác giả tiến hành tính toán sự chênh lệch trên *class recall* và *global model accuracy* khi không có tấn công và khi xảy ra tấn công đầu độc với m = 50% của tất cả các nhãn có trong tập dữ liệu huấn luyện theo *Công thức (2) và (3)*.

$$\Delta c^{recall} = c_{NP}^{recall} - c_{50}^{recall} \quad (2)$$

$$\Delta G^{acc} = G_{NP}^{acc} - G_{50}^{acc} \quad (3)$$

$c_{src} \rightarrow c_{target}$	Δc_{src}^{recall}	$\Delta c_{target}^{recall}$	$\sum all\ other\ \Delta c^{recall}$	ΔG^{acc}
F-MNIST				
1 → 3	0.74	-0.01	0.01	7.45%
4 → 6	0.63	-0.03	-0.06	5.36%
6 → 0	0.45	-0.09	-0.05	3.1%
MNIST				
3 → 9	0.61	-0.005	0.003	6.13%
6 → 0	0.77	-0.004	-0.01	7.21%
7 → 2	0.63	0.0	-0.005	6.42%

Bảng 6.3. Sự chênh lệch của *source class recall*, *target class recall* và *target class recall* của các lớp còn lại khi không có tấn công và khi xay ra tấn công.

Bảng 6.3 thể hiện sự chênh lệch của các thiết lập so sánh giữa khi không có tấn công và khi tỉ lệ kẻ tấn công đạt mức cao nhất (m = 50%) với 3 tiêu chí: sự mất mát trong lớp nguồn c_{src}, sự mất mát trong lớp mục tiêu c_{target}, sự mất mát của các lớp còn lại ($\sum c^{recall}$). Kết quả thực nghiệm chỉ răng có sự chênh lệch lớn của c_{src}^{recall} ở cả 2 tập dữ liệu *F-MNIST* và *MNIST* khi tấn công xảy ra. Tuy nhiên, giá trị chênh lệch của c_{target}^{recall} và $\sum c^{recall}$ là không đáng kể, cụ thể như sau:

- **Đối với tập dữ liệu *F-MNIST***, kết quả cho thấy sự chênh lệch giá trị c_{target}^{recall} ở thiết lập 6 → 0 là cao nhất và lớn hơn nhiều so với 2 thiết lập còn lại (≈ 0.08 và (≈ 0.06). Giá trị $\sum c^{recall}$ cũng cho thấy sự chênh lệch không đáng kể với giá trị chênh lệch rất thấp (≈ 0.06). Tuy nhiên, cả 2 giá trị c_{target}^{recall} và $\sum c^{recall}$ đều cho thấy xu hướng tăng kể cả khi tấn công với mức tỉ lệ *attacker* ở mức cao nhất.
- **Đối với tập dữ liệu *MNIST***, kết quả cho thấy xu hướng tương tự với tập dữ liệu *F-MNIST*. Tuy nhiên, các giá trị chênh lệch của c_{target}^{recall} và $\sum c^{recall}$ nhỏ hơn rất nhiều so với tập dữ liệu *F-MNIST*, cụ thể là chênh lệch ở mức cao nhất của c_{target}^{recall} và $\sum c^{recall}$ là 0.005. Các giá trị này cũng có xu hướng tăng khi có tấn công.

Giá trị ΔG^{acc} cho thấy khi có tấn công xảy ra thì sẽ giảm so với lúc không có tấn công.

Kết quả thực nghiệm cho rằng c_{target}^{recall} và $\sum c^{recall}$ (ngoài 2 lớp c_{src} và c_{target} có sự chênh lệch không đáng kể, hay thậm chí có xu hướng tăng khi tỉ lệ *attacker* đạt 50%. Qua đó, nhóm tác giả có thể chứng minh các đặc điểm sau của phương pháp *label-flipping attacks*:

- Tấn công không gây ảnh hưởng đến gía trị *class recall* của các lớp nhãn ngoại trừ c_{src}.
- Giá trị *global model accuracy* G^{acc} giảm là do ảnh hưởng chính của giá trị c_{src}^{recall}.
- Phương pháp *label-flipping attacks* là *targeted attack* vì chỉ có giá trị c_{src}^{recall} bị ảnh hưởng trong suốt quá trình xảy ra tấn công.

Để có sự nhìn nhận rõ hơn, nhóm tác giả sẽ tiếp tục phân tích phương pháp *label-flipping attacks* bằng cách sử dụng dữ liệu giả sinh ra từ mô hình *ACGAN*.

6.3.2. Cải tiến phương pháp tấn công đầu độc sử dụng Mạng đối nghịch tạo sinh (GAN)

6.3.2.1. Cấu hình thực nghiệm

Cấu hình quá trình huấn luyện (Training configurations): Ở phương pháp tấn công này, tác giả lựa chọn và cấu hình các thông số của quá trình FL cũng như các thông số của quá trình huấn luyện dựa vào thực nghiệm và phân tích trong [8]. Cụ thể như sau:

- *Tổng số người tham gia* (Pariticipants): Tổng số người tham gia vào quá trình *FL*, mặc định trong phương pháp tấn công này là 33.
- *Số người tham gia trong một vòng* (Participants per round): Số người tham gia trong 1 vòng do *central server* lựa chọn ngẫu nhiên, mặc định là 5.
- *Dữ liệu của người tham gia*: Dữ liệu huấn luyện của người tham gia sẽ bằng số lượng mẫu dữ liệu của từng tập dữ liệu *MNIST* và *F-MNIST* chia đều cho số người tham gia, cụ thể là 1818 mẫu dữ liệu.
- *Thông số của quá trình huấn luyện*: Người tham gia bình thường sẽ có *epoch* là 1 và *lr* là 0.001. Trong khi đó, *attacker* lần lượt có epoch = 5 và lr = 0.002 (cao gấp đôi so với người tham gia bình thường).

- *Nhiệm vụ của Attacker*: Trong suốt quá trình *FL, attacker* có nhiệm vụ huấn luyện mô hình *ACGAN*. Khi mô hình *ACGAN* đạt được hiệu quả nhất định (độ chính xác > 90%) sẽ thực hiện sinh dữ liệu tấn công mô hình *FL*.

Cấu hình kĩ thuật tấn công (Attack configurations): Ở phương pháp tấn công này, nhóm tác giả tiếp tục sử dụng kĩ thuật *label-flipping* để triển khai tấn công đầu độc dữ liệu. Tuy nhiên, dữ liệu đầu độc được sinh ra từ mô hình *ACGAN* khác với dữ liệu trong *Phương pháp A*. Nhóm tác giả sẽ tiến hành đưa ra các giả thiết khác nhau về các khả năng của *attacker* để có cái nhìn toàn diện hơn. Các kịch bản thực nghiệm cụ thể sẽ được tác giả mô tả ở phần sau.

Trong các phương pháp tấn công, mục tiêu và khả năng của *attacker* ảnh hưởng đến kết quả cũng như quá trình tấn công theo các trường hợp khác nhau. Đối với mỗi trường hợp tấn công khác nhau, *attacker* sẽ có những phương pháp và cách thức tấn công khác nhau. Để làm rõ từng trường hợp của *attacker*, nhóm tác giả sẽ phân tích và đưa ra 2 trường hợp khả năng của *attacker* có thể xảy ra trong thực tế. Cụ thể gồm các kịch bản thực nghiệm như sau:

1. *Attacker* có dữ liệu thuộc đầy đủ các nhãn từ người dùng khác.
2. *Attacker* không có dữ liệu thuộc các nhãn từ người dùng khác.

6.3.2.2. Kịch bản B1 - Attacker có dữ liệu thuộc đầy đủ các nhãn từ người dùng khác

Ở kịch bản này, nhóm tác giả sẽ giả sử rằng *attacker* chiếm được một người tham gia có đầy đủ các nhãn dữ liệu dùng để huấn luyện trong quá trình FL. Do đó, *attacker* có đầy đủ các nhãn dữ liệu, điều này hỗ trợ cho quá trình huấn luyện mô hình *ACGAN* được triển khai cục bộ. Khi mô hình *ACGAN* được huấn luyện trên tất cả các nhãn dữ liệu, thì *Generator* có thể sinh dữ liệu thuộc lớp nhãn bất kì để thực hiện tấn công đầu độc. Trong thực tế, trường hợp nhóm tác giả đưa ra là khả thi vì kẻ tấn công có thể sử dụng các kĩ thuật tấn công để chiếm được 1 thiết bị tham gia vào quá trình *FL*.

Attacker sẽ thực hiện tấn công đầu độc dữ liệu chống lại mô hình *FL* bằng kĩ thuật *label-flipping* giống với *Phương pháp A* đã được đề cập ở trên. Tuy nhiên, trong trường hợp này dữ liệu đầu độc được sinh ra từ mô hình *ACGAN* sau khi đã đạt được một mức độ hiệu quả nhất định. *Attacker* tiến hành chọn lớp nhãn nguồn c_{src}, sau đó sử dụng *Generator* để sinh dữ liệu thuộc lớp nhãn c_{src} với một số lượng mẫu dữ liệu đủ để cuộc tấn công đạt hiệu quả và có ảnh hưởng. *Attacker* thực hiện tấn công chỉ

một lần duy nhất (single-round attack) và áp dụng hệ số khuếch đại (scale factor) để làm tăng mức độ ảnh hưởng của cuộc tấn công và làm cho lớp dữ liệu c_{src} phân loại sai ở các vòng tiếp theo. Mục đích chính của *attacker* là làm cho mô hình phân loại sai lớp nhãn c_{src} thành c_{target} (được *attacker* chọn) tại vòng xảy ra tấn công và gây ảnh hưởng đến các vòng sau đó. Cụ thể, trong kịch bản này tác giả lựa chọn cặp nhãn dữ liệu cho 2 tập dữ liệu *MNIST* và *F-MNIST* là 0 → 3 và 4: coat →6: t-shirt.

Generator sẽ thực hiện sinh dữ liệu thuộc lớp nhãn 0 và 4: coat, sau đó thay đổi nhãn dữ liệu thành 3 và 6: t-shirt để tạo dữ liệu đầu độc mô hình.

a) Sinh dữ liệu tấn công sử dụng mô hình ACGAN

Việc sinh dữ liệu tấn công sử dụng mô hình *ACGAN* rất quan trọng và gây ảnh hưởng đến kết quả của cuộc tấn công. Hiệu quả việc sinh dữ liệu của *Generator* phụ thuộc vào cấu trúc mô hình và số lần huấn luyện mô hình *ACGAN*. Để dữ liệu sinh ra gần giống thật thì *attacker* phải huấn luyện *ACGAN* nhiều lần trước khi sinh dữ liệu tấn công và đảm bảo được dữ liệu sinh ra phải được *global model* phân loại thành lớp nhãn c_{src} mà *attacker* đã chọn trước đó. Việc sinh dữ liệu tấn công gần giống với dữ liệu thật nhất sẽ có ảnh hưởng tốt tới kết quả tấn công, ngược lại sẽ gây ảnh hưởng đến các nhãn dữ liệu không bị tấn công.

(a) 300 iteration (b) 500 iteration (c) Real samples

Hình 6.3. Kết quả việc sinh dữ liệu dựa trên mô hình GAN của tập dữ liệu MNIST.

(a) 500 iteration (b) 700 iteration (c) Real samples

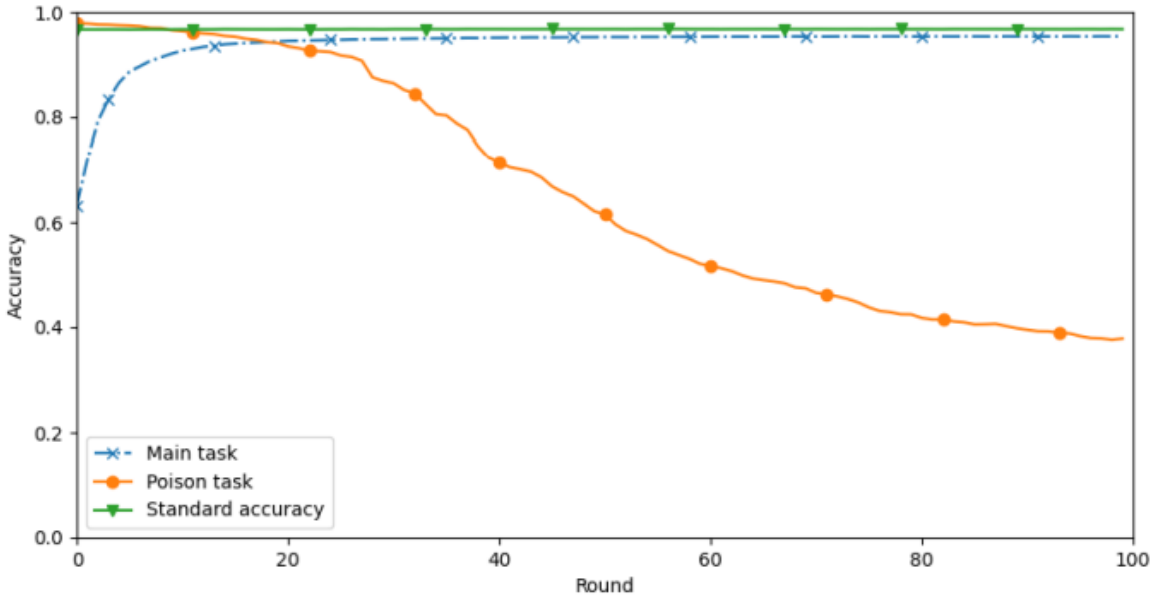
Hình 6.4. Kết quả việc sinh dữ liệu dựa trên mô hình GAN của tập dữ liệu FMNIST.

Hình 6.3 và 6.4 lần lượt mô tả kết quả của dữ liệu được tạo ra từ mô hình *ACGAN* cho cả 2 tập dữ liệu *MNIST* và *F-MNIST*. Nhóm tác giả tiến hành thực nghiệm việc tạo dữ liệu của *ACGAN* qua 500 lần chạy (iteration) đối với tập dữ liệu MNIST và 700 iteration đối với F-MNIST. Như đã thể hiện trong *Hình 6.3.a) và 6.3.b)*, việc tái xây dựng dữ liệu từ mô hình *ACGAN* qua 300 và 500 *iteration* đã cho thấy kết quả gần giống với dữ liệu thật của tập dữ liệu *MNIST* và *F-MNIST*. Qua 500 và 700 *iteration*, kết quả cho thấy dữ liệu giả sinh ra càng giống với dữ liệu thật hơn khi mà *Generator* trở nên tốt hơn qua từng round. Dựa trên hình ảnh mô phỏng dữ liệu giả đã sinh ra từ mô hình *ACGAN* nhóm tác giả có thể đưa ra kết luận rằng việc triển khai mô hình *ACGAN* trong môi trường FL là khả thi và đạt được mức độ hiệu quả nhất định. Vì thế, bằng cách xây dựng mô hình *ACGAN*, kẻ tấn công có thể tạo ra dữ liệu giả giống với dữ liệu của các người tham gia khác.

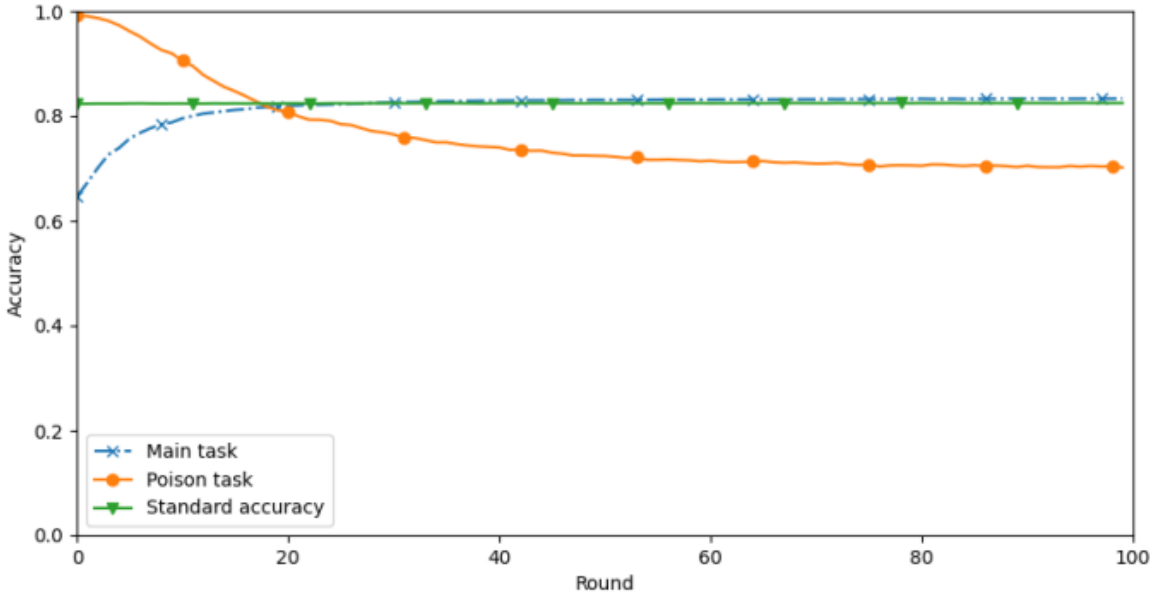
b) Thực nghiệm B1.1: Đánh giá kết quả của cuộc tấn công qua từng vòng

Để đánh giá mức độ ảnh hưởng của cuộc tấn công đến *Main task* và *Poison task* qua từng round sau khi thực hiện tấn công, nhóm tác giả thực nghiệm *label-flipping attacks* dựa trên dữ liệu được sinh ra từ mô hình *ACGAN*. Đồng thời, nhóm tác giả đánh giá sự ảnh hưởng của dữ liệu sinh ra từ mô hình *ACGAN* đến kết quả của cuộc tấn công khi dữ liệu sinh ra từ *Generator* chưa đạt được độ chính xác cao.

Nhóm tác giả phân tích dựa trên kết quả của giá trị *Main task* và *Poison task* qua 100 vòng sau khi kết thúc tấn công. Tiến hành thực nghiệm trong 3 lần đối với mỗi trường hợp sau đó tính toán giá trị trung bình để đưa ra giá trị *Main task* và *Poison task* qua từng vòng.



(a) MNIST



(b) F-MNIST

Hình 6.5. Độ chính xác của Poison task và Main task của 2 tập dữ liệu *MNIST* và *F-MNIST* qua từng vòng.

Kết quả thực nghiệm tại *Hình 6.5* cho thấy kết quả của *Main task* và *Poison task* qua 100 vòng sau khi thực hiện tấn công, trong đó đường *Standard accuracy* biểu thị cho độ chính xác của *global model* khi không có tấn công xảy ra. Kết quả thực nghiệm thu thập với cách thiết lập giá trị chỉ số *scale factor* bằng nhau ($S = 7$) cho cả 2 tập dữ liệu *MNIST* và *F-MNIST*. Giá trị *Poison task* trong cả 2 thực nghiệm cho thấy xu

hướng giảm khi càng về sau quá trình FL; mặt khác, giá trị *Main task* sau khi giảm tại vòng xảy ra tấn công thì có xu hướng tăng lên bằng với *Standard accuracy*. Cụ thể:

- **Kết quả thực nghiệm trên tập dữ liệu *MNIST*** mô tả ở *Hình 6.5.a)* cho thấy rằng tại vòng xảy ra tấn công đầu độc, giá trị *Main task accuracy* và *Poison task accuracy* lần lượt là xấp xỉ 62% và 99%. Các giá trị này có xu hướng thay đổi khi số vòng sau tấn công tăng lên. Tại round thứ 100 sau khi kết thúc tấn công thì giá trị *Main task accuracy* $\approx 97\%$ (tăng 35% so với vòng xảy ra tấn công) và *Poison task accuracy* $\approx 39\%$ (giảm 60% so với vòng xảy ra tấn công). Qua kết quả của *Poison task* cho thấy ảnh hưởng của cuộc tấn công có thể duy trì lâu và tỉ lệ tấn công thành công vẫn đạt được ở mức ổn định (> 40%).
- **Trong khi đó, kết quả thực nghiệm trên tập dữ liệu *F-MNIST*** (Hình 6.5.b) cũng cho thấy xu hướng tương tư với tập dữ liệu *MNIST*. Tại vòng xảy ra tấn công, giá trị *Main task accuracy* $\approx 65\%$ và *Poison task accuracy* $\approx 100\%$, các giá trị này thay đổi theo từng vòng cho đến vòng thứ 100 sau khi xảy ra tấn công thì có giá trị lần lượt là $\approx 82\%$ (tăng 17%) và 70% (giảm 31%). Ảnh hưởng của cuộc tấn công đến *global model* được duy trì và tỉ lệ tấn công đạt ở mức cao (> 70%).

Tác giả có thể giải thích cho sự thay đổi của các giá trị *Main task* và *Poison task* rằng do sau khi xảy ra tấn công đầu độc, khi quá trình *FL* tiếp diễn thì các cập nhật của người tham gia bình thường sẽ liên tục được gửi đến cho *central server* để tổng hợp mô hình. Điều này làm cho cập nhật độc hại của *attacker* dần bị triệt tiêu làm cho giá trị của *Poison task* giảm. Bên cạnh đó, giá trị *Main task accuracy* tăng dần trở lại với mức bình thưởng (so sánh với *Standard accuracy*).

Qua kết quả thực nghiệm trong *Hình 6.5* cũng cho thấy sự chênh lệch các giá trị *Main task accuracy* và *Poison task accuracy* của 2 tập dữ liệu *MNIST* và *F-MNIST*. Lí do gây ra sự chênh lệch lớn này là do dữ liệu tấn công sinh ra từ mô hình *GAN* của tập dữ liệu *F-MNIST* chính xác hơn so với tập dữ liệu *MNIST*. Tuy nhiên, chỉ số *scale factor* cũng ảnh hưởng trực tiếp đến mức độ duy trì và tỉ lệ tấn công thành công của cuộc tấn công đối với từng tập dữ liệu. Nhóm tác giả sẽ có những phân tích và đánh

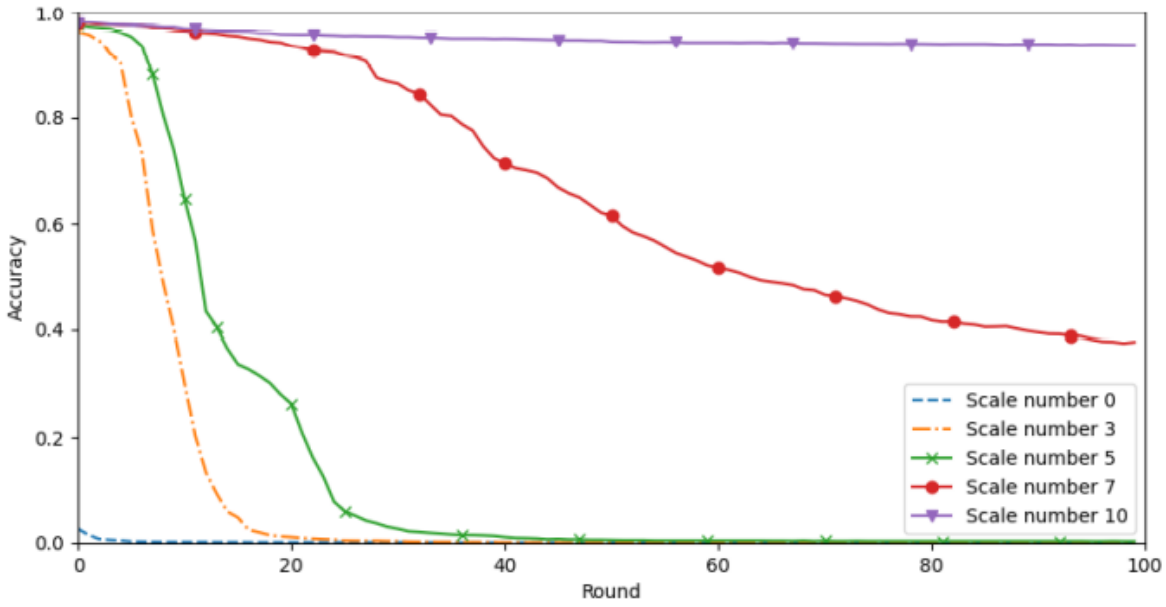
giá để làm rõ mức độ ảnh hưởng của chỉ số *scale factor* đến kết quả của cuộc tấn công trong thực nghiệm sau.

Kết quả trên cho thấy việc thực hiện tấn công đầu độc dữ liệu chỉ trong 1 round duy nhất đạt được hiệu quả cao và duy trì được ảnh hưởng của cuộc tấn công phụ thuộc vào chỉ số *scale factor* mà *attacker* chọn. Bên cạnh đó, tấn công làm ảnh hưởng đến *Main task accuracy* nhưng giá trị này dần cải thiện lại với mức ổn định ban đầu.

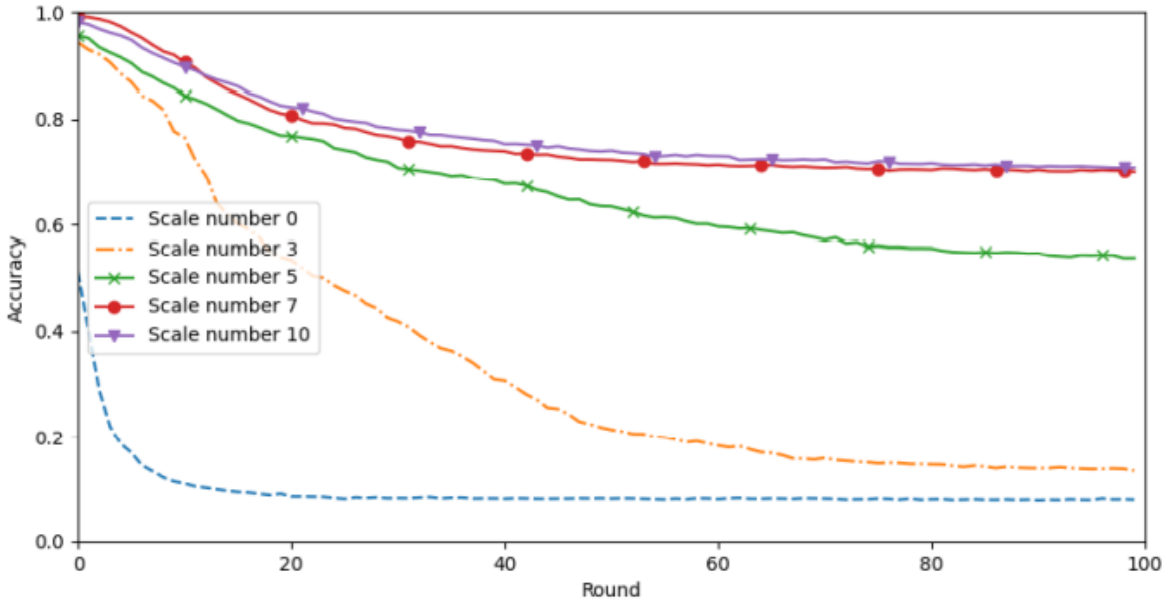
c) Thực nghiệm B1.2: So sánh và đánh giá mức độ duy trì của cuộc tấn công với các chỉ số scale factor khác nhau

Mức độ duy trì của phương pháp tấn công *single-round* phụ thuộc sự thay đổi của chỉ số *scale factor* mà *attacker* sử dụng. Để có cái nhìn khách quan hơn về ảnh hưởng của *scale factor* đến hiệu quả của cuộc tấn công, tác giả tiến hành thực nghiệm tấn công đầu độc với các chỉ số *scale factor* khác nhau trên 2 tập dữ liệu *MNIST* và *F-MNIST*. Từ đó đánh giá mức độ ảnh hưởng của cuộc tấn công theo từng chỉ số.

Tiến hành phân tích giá trị *Poison task* của từng thực nghiệm với các chỉ số *scale factor* khác nhau sau khi xảy ra tấn công trong 100 rounds. Theo dõi sự thay đổi giá trị *Poison task* sau khi tấn công và số round mà phương pháp tấn công duy trì được.



(a) MNIST



(b) F-MNIST

Hình 6.6. Mức độ ảnh hưởng của chỉ số Scale factor đến sự duy trì và tỉ lệ tấn công thành công của cuộc tấn công.

Mục đích chính chỉ số *scale factor* (S) là làm tăng mức độ ảnh hưởng cập nhật mô hình của *attacker* đến các tham số của *global model*. Để có sự nhìn nhận khách quan về kết quả thực nghiệm, tác giả tiến hành chạy và đánh giá kết quả của 100 rounds tiếp theo sau khi xảy ra tấn công dưới 5 giá trị *scale factor* khác nhau (S = 0, 3, 5, 7, 10). *Hình 6.6* mô tả kết quả của giá trị *Poison task* với các chỉ số *scale factor* khác

nhau cho 2 tập dữ liệu *MNIST* và *F-MNIST*. Nhìn chung, có thể thấy giá trị *S* gây ảnh hưởng đến tỉ lệ tấn công thành công của cuộc tấn công trên cả 2 tập dữ liệu. Cụ thể:

- **Kết quả thực nghiệm của tập dữ liệu *MNIST*** được mô tả ở *Hình 6.6.a* cho thấy với mỗi giá trị *S* khác nhau sẽ gây ảnh hưởng đến tỉ lệ tấn công thành công với mức độ khác nhau. Có thể thấy rằng với mọi chỉ số *S* thì tỉ lệ tấn công thành công tại vòng xảy ra tấn công xấp xỉ nhau. Với chỉ số *S* cao (S = 7, 10) cho thấy tỉ lệ tấn công thành công vẫn duy trì ở mức ổn định sau khi xảy ra tấn công 100 rounds, và tỉ lệ này giữ nguyên ở trạng thái cao với S = 10. Với *S* lần lượt là 3 và 5 thì *Poison task* chỉ duy trì đến được vòng thứ 20 (S = 3) và 40 (S = 5) sau khi tấn công. Khi cập nhật mô hình cục bộ của *attacker* không được mở rộng (scale-up) thì cuộc tấn công không gây ảnh hưởng đến *global model*.
- **Trong khi đó, kết quả thực nghiệm của tập dữ liệu *F-MNIST*** (mô tả ở *Hình 6.6.b*) cho thấy xu hướng giá trị *Poison task* giảm tương tự như kết quả thực nghiệm trên tập dữ liệu *MNIST*. Tỉ lệ tấn công thành công được duy trì cho đến 100 vòng sau khi tấn công với mọi chỉ số *S*. Tuy nhiên, tỉ lệ tấn công thành công tại vòng xảy ra tấn công có sự chênh lệch (nhưng không đáng kể) đối với các giá trị S khác nhau. Với S = {7, 10}, *Poison task* cho thấy xu hướng giảm qua từng vòng sau khi tấn công, nhưng với S = 10 thì tỉ lệ tấn công thành công của tập dữ liệu *MNIST* chỉ giảm với tỉ lệ rất thấp. Nhìn chung, kết quả thực nghiệm cho thấy giá trị *Poison task* được duy trì trong vòng 100 vòng sau khi tấn công và tỉ lệ tấn công thành công được duy trì ở mức ổn định (đối với S = {5, 7, 10}) và mức thấp (S = {0 ,3}).

Với mỗi giá trị *scale factor* khác nhau làm cho cuộc tấn công đạt hiệu quả khác nhau về cả *Poison task* và *Main task*. Bên cạnh đó, do *attacker* huấn luyện dữ liệu đầu độc với số lượng lớn *epoch* nên cũng làm ảnh hưởng đến độ chính xác của *global model*.

Qua kết quả thực nghiệm cho thấy hiệu quả việc tấn công đầu độc dữ liệu chỉ trong 1 round duy nhất phụ thuộc vào giá trị của *scale factor*. Tuy nhiên, việc lựa chọn giá trị *S* phù hợp cũng rất quan trọng để cho *central server* không phát hiện ra những bất thường trong mô hình và không gây ảnh hưởng đến giá trị của *Main task*.

6.3.2.3. Kịch bản B2 - Attacker không có dữ liệu thuộc các nhãn từ người dùng khác

Trong trường hợp xấu nhất, *attacker* không thể chiếm được bất kì người tham gia nào trong quá trình *FL*. Từ đó nảy sinh ra những thách thức về dữ liệu để huấn luyện mô hình *ACGAN* sinh ra dữ liệu đầu độc. Tác giả đã đưa ra trường hợp *attacker* không biết dữ liệu huấn luyện của các người tham gia còn lại, khi dữ liệu bị hạn chế thì kĩ thuật tấn công sẽ thay đổi như thế nào.

Giả sử rằng *attacker* không chiếm được quyền kiểm soát của một người tham gia vào quá trình FL, khi đó dữ liệu huấn luyện sẽ bị hạn chế để huấn luyện mô hình *ACGAN* sinh dữ liệu. Trong kịch bản này, tuy *attacker* không có đầy đủ dữ nhãn dữ liệu huấn luyện, nhưng lại có một lớp nhãn dữ liệu mà tất cả các người tham gia khác không biết đến và số lượng mẫu dữ liệu lớp nhãn này là hạn chế. *Attacker* sẽ huấn luyện mô hình *ACGAN* trên nhãn dữ liệu này, việc này làm cho *Generator* chỉ sinh ra được dữ liệu thuộc nhãn mà *attacker* nắm giữ. Kịch bản xây dựng này phù hợp với thực tế hơn so với kịch bản trước, tuy nhiên lại có những hạn chế và thay đổi rõ ràng trong kĩ thuật sinh dữ liệu tấn công.

Giống như các thực nghiệm trước đó, trong kịch bản này *attacker* cũng sẽ thực hiện tấn công đầu độc dữ liệu bằng kĩ thuật *label-flipping*. Nhược điểm của *Generator* là chỉ sinh được dữ liệu giống với dữ liệu của *attacker*. Tác giả đưa ra giả thiết rằng trong suốt quá trình *FL attacker* không thực hiện huấn luyện mô hình cục bộ trên tập dữ liệu đó và không gửi những cập nhật đến cho *central server* thì *global model* sẽ không học được những đặc trưng của lớp dữ liệu mà *attacker* nắm giữ. Vì vậy, *attacker* có thể xây dựng tập dữ liệu đầu độc bằng cách lợi dung sự phân loại của *global model* đối với từng mẫu dữ liệu sinh ra từ *Generator*. Cụ thể, với mỗi mẫu dữ liệu x_{fake} được sinh ra từ *Generator*, nếu *global model* phân loại $G_{x_{fake}} = c_{src}$ thì *attacker* sẽ thực hành thay đổi nhãn dữ liệu của c_{src} thành c_{target}, với c_{src} và c_{target} là lớp nhãn được *attacker* chọn và nằm trong tập nhãn của các người dùng tham gia khác. Trong kịch bản này, *attacker* thực hiện tấn công trong nhiều vòng sau khi mô hình *ACGAN* đạt được độ hiệu quả nhất định (multiple-round attacks) với số lượng người tham gia trong 1 vòng FL thay đổi tăng dần.

a) Sinh dữ liệu tấn công sử dụng mô hình ACGAN

Với kịch bản này, tác giả đặt trường hợp *attacker* sẽ nắm một lớp nhãn dữ liệu mà tất cả các người tham gia khác không có dữ liệu thuộc lớp nhãn này, gọi là $c_{attacker}$. Cụ thể như sau:

- **MNIST**: *Attacker* sẽ nắm giữ lớp nhãn 7 và không người tham gia nào có dữ liệu thuộc nhãn 7. Tác giả chọn thiết lập $c_{src} \rightarrow c_{target}$ là 9 → 0.
- **F-MNIST**: *Attacker* nắm giữ lớp nhãn 6: shirt và chọn thiết lập 4: coat → 0: t-shirt/top.

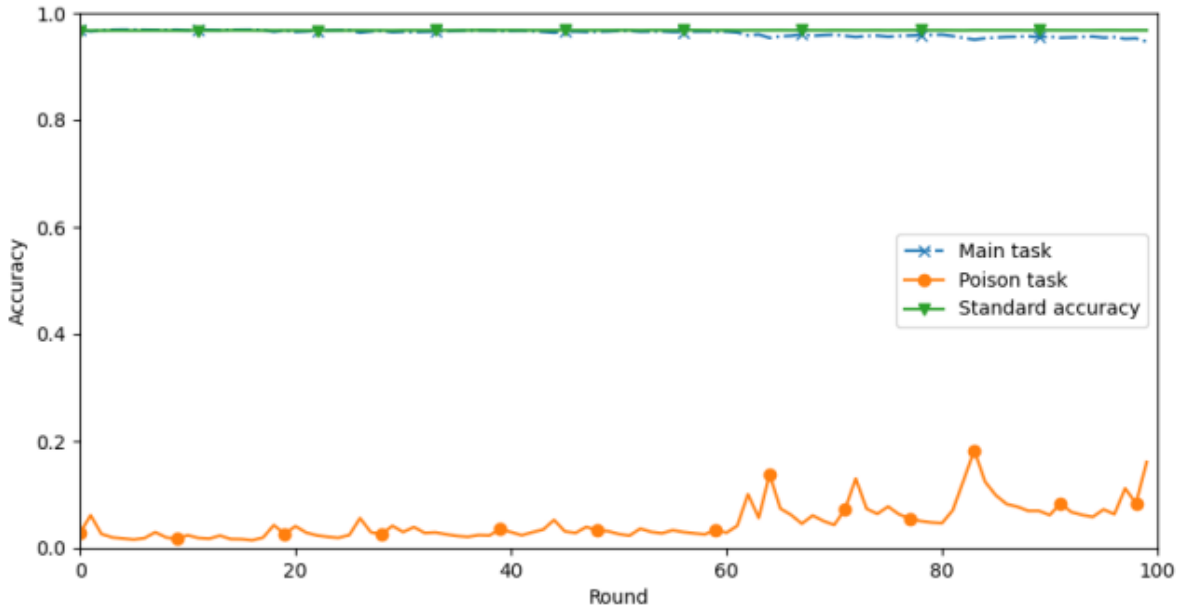
Mục đích của *attacker* là không tham gia vào quá trình huấn luyện *global model* nhằm không cho *global model* học được những đặc trưng của lớp nhãn $c_{attacker}$. *Attacker* chỉ tham gia huấn luyện *global model* khi mô hình *ACGAN* ổn định và sinh ra dữ liệu tấn công nhằm mục đích làm cho *global model* phân loại sai c_{src} thành c_{target}.

Generator chỉ sinh được dữ liệu giống với lớp nhãn mà *attacker* đang nắm giữ, không phải sinh dữ liệu giống với lớp nhãn c_{src}. Do đó, nếu *attacker* muốn tấn công đầu độc lớp nhãn c_{src} thì sẽ dùng dữ liệu thuộc lớp nhãn $c_{attacker}$ được *global model* phân loại thành c_{src}.

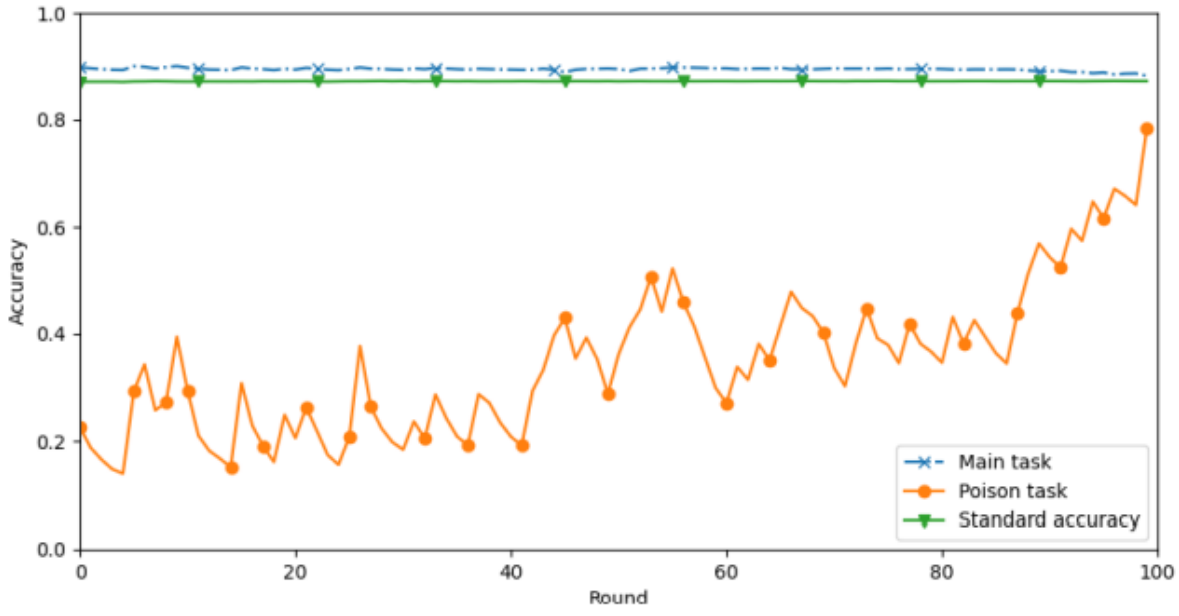
b) Thực nghiệm B2.1: Đánh giá kết quả của tấn công qua từng vòng khi duy trì tấn công đầu độc

Trong kịch bản này, nhóm tác giả sử dụng phương pháp *multiple-round attacks* để tiến hành thực nghiệm nhằm đánh giá được mức độ ảnh hưởng của phương pháp tấn công nếu được duy trì. Qua đó, đánh giá các giá trị *Main task* và *Poison task* khi phương pháp tấn công được duy trì bởi *attacker*.

Tiến hành thực nghiệm trên 2 tập dữ liệu *MNIST* và *F-MNIST*, triển khai tấn công trong vòng 100 rounds tiếp theo sau khi *attacker* lần đầu tiên tấn công đầu độc *global model* và theo dõi các giá trị *Main task* và *Poison task* qua từng round.



(a) MNIST



(b) F-MNIST

HÌnh 6.7. Hiệu quả của phương pháp multiple-round attacks qua từng round.

Hình 6.7 mô tả kết quả thực nghiệm của phương pháp *multiple-round attacks* được theo dõi trong vòng 100 rounds sau khi lần tấn công đầu tiên xảy ra với số người tham gia trong 1 vòng là 5. Kết quả cho thấy giá trị *Main task* không bị ảnh hưởng khi so so với giá trị *Standard accuracy*. Ngược lại, giá trị *Poison task* cho thấy 2 xu hướng khác nhau của 2 tập dữ liệu *MNIST* và *F-MNIST*, cụ thể như sau:

- **Đối với tập dữ liệu *MNIST***, tỉ lệ tấn công thành công rất thấp, chỉ duy trì được mức độ > 20% trong vòng 100 rounds và không gây ra ảnh hưởng đáng kể đến *global model*.
- **Đối với tập dữ liệu *F-MNIST***, giá trị *Poison task* thay đổi qua từng round và có xu hướng tăng dần khi duy trì tấn công trong vòng 100 rounds. Cụ thể ở round tấn công đầu tiến, giá trị *Poison task* ở mức ≈ 20% nhưng sau 100 rounds tiếp theo thì giá trị này đạt ≈ 79% (tăng ≈ 59%).

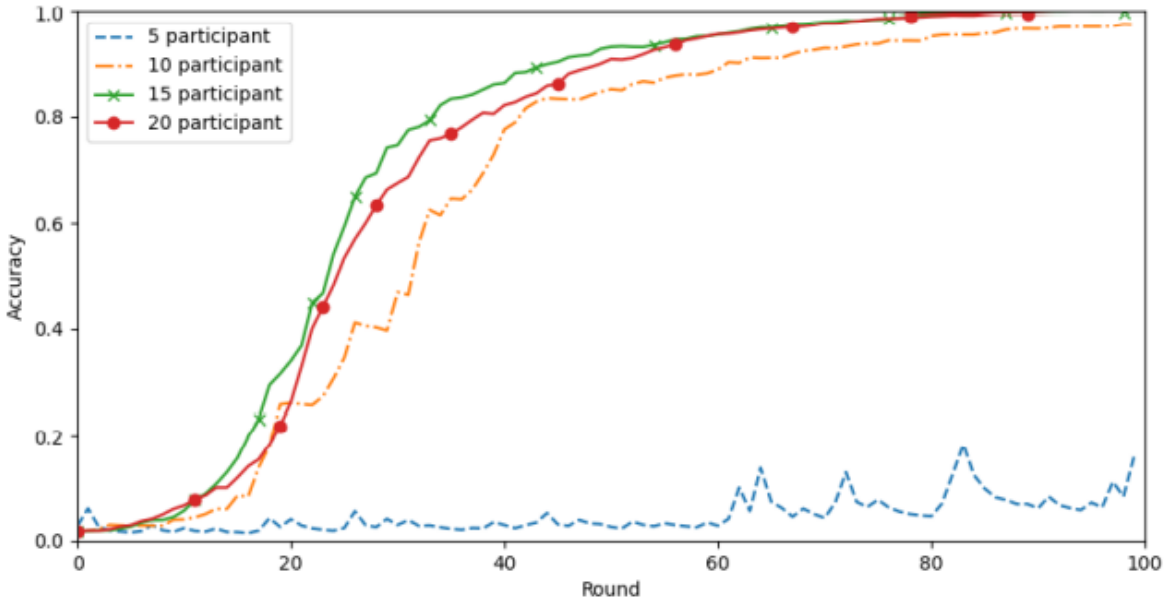
Lí do gây ra sự khác nhau về xu hướng giá trị *Poison task* của 2 tập dữ liệu là do tập dữ liệu *MNIST* đã đạt được độ chính xác rất cao khi không có tấn công (≈ 98%) nên khi thực hiện tấn công thì những cập nhật độc hại của *attacker* không gây ảnh hưởng nhiều. Mặt khác, tập dữ liệu *F-MNIST* có độ phức tạp cao hơn *MNIST* nên độ chính xác của mô hình thấp hơn khi không có tấn công, do đó khi có tấn công thì những cập nhật độc hại sẽ có ảnh hưởng lớn đến *global model*.

Kết quả thực nghiệm cho thấy rằng phương pháp *multiple-round attacks* tuy có ảnh hưởng đến giá trị *Poison task* nhưng phụ thuộc vào độ phức tạp của từng tập dữ liệu. Bên cạnh đó, số lượng người tham gia trong 1 round (participants per round - *PPR*) cũng ảnh hưởng đến hiệu quả của phương pháp tấn công. Để làm rõ sự ảnh hưởng này, tác giả sẽ tiến hành thực nghiệm với các giá trị *PPR* khác nhau.

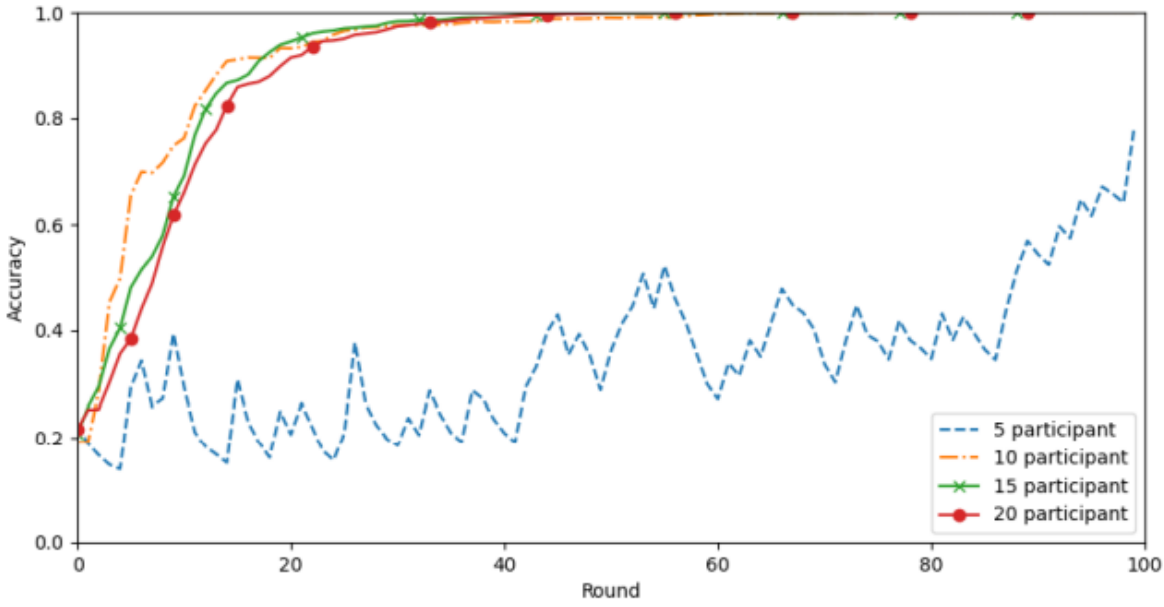
c) Thực nghiệm B2.2: Đánh giá sự thay đổi của Main task và Poison task khi tăng số lượng người tham gia trong một vòng

Để đánh giá sự thay đổi của các giá trị *Main task* và *Poison task* qua các vòng khi triển khai phương pháp *multiple-round attacks*, tác giả tiến hành thực nghiệm theo dõi các giá trị *Main task* và *Poison task* với giá trị *PPR* tăng dần. Qua đó đánh giá mức độ ảnh hưởng của từng giá trị *PPR* và tỉ lệ giữa giá trị *PPR* và tổng số *Participants* tham gia vào quá trình FL.

Tiến hành phân tích giá trị *Main task* và *Poison task* của từng thực nghiệm với các giá trị *PPR* khác nhau (PPR = {5, 10, 15, 20}) trong vòng 100 rounds với phương pháp *multiple-round attacks*. Tác giả gọi tỉ lệ giữa giá trị *PPR* và tổng số người tham gia là *Participants Ratio - PR*.



(a) MNIST



(b) F-MNIST

Hình 6.8. Sự thay đổi của giá trị Poison task với các chỉ số PPR khác nhau.

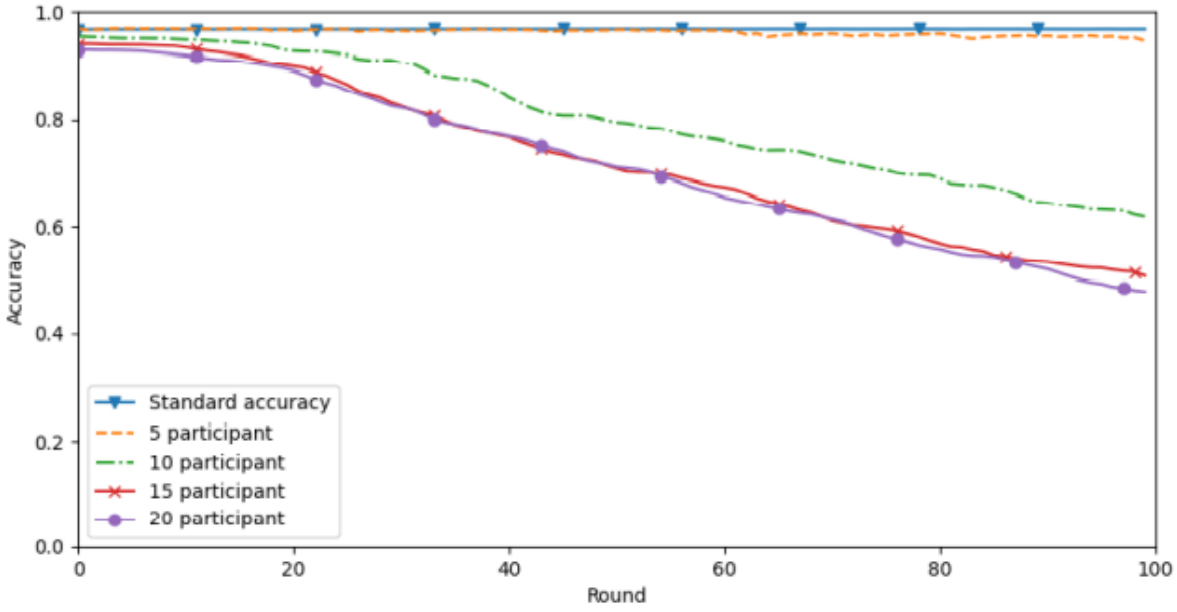
Hình 6.8 mô tả kết quả thực nghiệm giá trị *Poison task* qua từng vòng với mỗi giá trị *PPR* khác nhau cho cả 2 tập dữ liệu *MNIST* và *F-MNIST*. Có thể thấy rằng giá trị *Poison task* đều có xu hướng tăng cho cả 2 tập dữ liệu qua từng vòng khi duy trì tấn công trong vòng 100 rounds tiếp theo. Cụ thể như sau:

- **Đối với tập dữ liệu *MNIST***, giá trị *Poison task* tại round tấn công đầu tiên rất thấp nhưng có xu hướng tăng dần qua từng vòng. Với PPR = 5 thì tấn

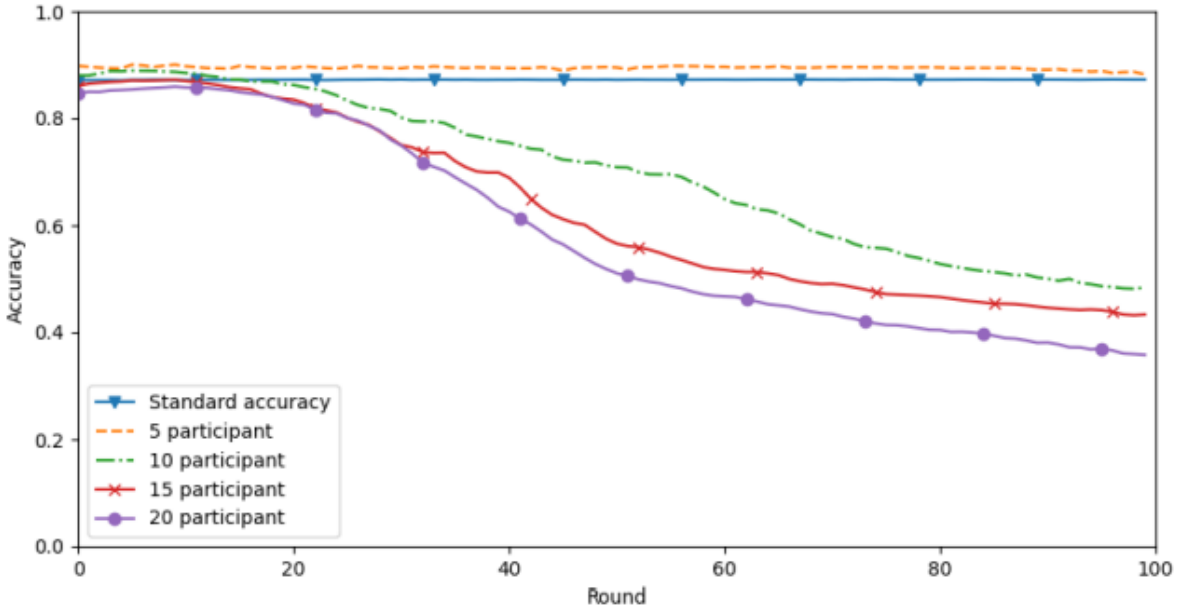
công gây ảnh hưởng rất thấp, với PPR = {15, 20} thì tỉ lệ tấn công thành công tăng dần qua từng rounds, cụ thể từ ≈ 1% đến ≈ 100%. Tỉ lệ tấn công thành công từ round thứ 30 trở đi đạt mức ổn định (> 50%) và đạt mức cao (> 80%) từ round 40 đối với các giá trị PPR = {10, 15, 20}.

- **Đối với tập dữ liệu *F-MNIST***, tỉ lệ tấn công thành công tại round đầu tiên cao hơn so với *MNIST* (≈ 20%). Các giá trị PPR = {10, 15, 20} đều cho thấy *Poison task* có xu hướng tăng dần theo từng vòng và đạt đến mức cao nhất, tại round thứ 20 đã cho thấy tỉ lệ tấn công thành công cao (≈ 90%).

Ảnh hưởng của cuộc tấn công phụ thuộc vào sự thay đổi của tỉ lệ *PPR* vì giá trị *PPR* cao thì tỉ lệ *attacker* được chọn vào những round liên tiếp nhau sẽ tăng. Từ đó, *attacker* có thể thường xuyên đầu độc *global model* bằng dữ liệu sinh ra từ mô hình *ACGAN*.



(a) MNIST



(b) F-MNIST

Hình 6.9. Sự thay đổi của giá trị Main task với các chỉ số PPR khác nhau.

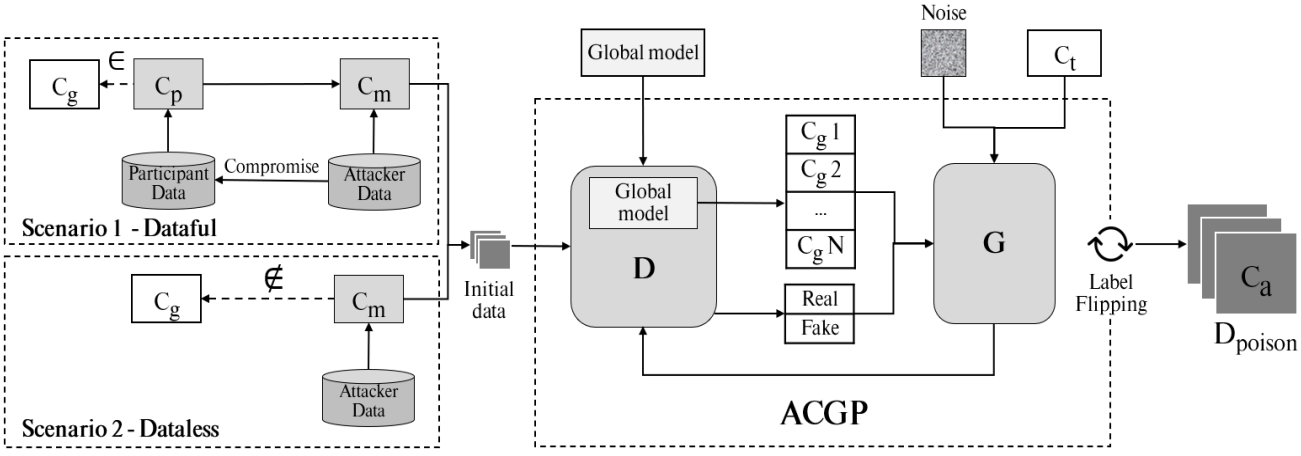
Hình 6.9 mô tả kết quả thực nghiệm của giá trị *Main task* qua từng vòng với mỗi giá trị *PPR* khác nhau cả 2 tập dữ liệu *MNIST* và *F-MNIST*. Khi PPR = 5 thì không gây ảnh hưởng đến giá trị *Main task* (so sánh với giá trị *Standard accuracy*). Tuy nhiên, với PPR = {10, 15, 20} thì giá trị *Main task* có xu hướng giảm dần qua từng round, cụ thể như sau:

- **Đối với tập dữ liệu *MNIST***, *Main task* giảm từ ≈ 95% xuống ≈ 50% với PPR = {15, 20}, cụ thể giảm≈ 45%.
- **Đối với tập dữ liệu *F-MNIST***, giá trị *Main task* cho thấy xu hướng tương tự với tập dữ liệu *MNIST* khi có dấu hiệu giảm mạnh sau khi kết thúc 100 rounds, cụ thể là giảm từ *85%* xuống > 40% với PPR = 20.

Giá trị *Main task* tương ứng với độ chính xác của *global model* nên khi giá trị này giảm thì *global model* đã không đạt được độ chính xác cao ở các lớp nhãn không bị tấn công. Cụ thể ở trong trường hợp này, các lớp nhãn không bị tấn công đã bị ảnh hưởng dẫn đến độ chính xác của *global model* giảm, điều này làm mất đi một đặc điểm của phương pháp *label-flipping attacks* là tấn công có mục tiêu. Tác giả có thể giải thích cho trường hợp này là vì dữ liệu sinh ra từ mô hình *ACGAN* không đạt được độ chính xác cao, hay nói cách khác là khi quá trình *FL* càng kéo dài thì những dữ liệu được sinh ra này sẽ không còn phù hợp để tấn công.

Thực nghiệm cho thấy hiệu quả của phương pháp *multiple-round attacks* cũng phụ thuộc vào giá trị *PPR* khi mà *attacker* được chọn thường xuyên hơn sẽ làm cho *global model* dễ bị đầu độc hơn. Khi giá trị *PPR* tăng, khả năng *attacker* sẽ thực hiện đầu độc mô hình cao hơn dẫn đến giá trị *Main task* giảm (vì gây ảnh hưởng đến các lớp nhãn không bị tấn công) và giá trị *Poison task* tăng (đúng với kĩ thuật *label-flipping*).

7. Hình ảnh, sơ đồ minh họa chính



Hình 7.1. Tổng quát mô hình và các kịch bản tấn công

8. Tài liệu tham khảo

[1] L. Lyu, H. Yu, J. Zhao, and Q. Yang, "Threats to Federated Learning," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12500 LNCS, pp. 3–16, 2020, doi: 10.1007/978-3-030-63076-8_1.

[2] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, vol. 54, 2017.

[3] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, and B. He, "Federated learning systems: vision, hype and reality for data privacy and protection," *arXiv*, pp. 1–46, 2019.

[4] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12308 LNCS, no. 2, pp. 480–501, 2020, doi: 10.1007/978-3-030-58951-6_24.

[5] G. Sun, Y. Cong, J. Dong, Q. Wang, and J. Liu, "Data poisoning attacks on federated machine learning," *arXiv*, vol. 14, no. 8, pp. 1–8, 2020.

[6] J. Zhang, J. Chen, D. Wu, B. Chen, and S. Yu, "Poisoning attack in federated learning using generative adversarial nets," *Proceedings - 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering, TrustCom/BigDataSE 2019*, no. August, pp. 374–380, 2019, doi: 10.1109/TrustCom/BigDataSE.2019.00057.

[7] L. Muoz-Gonzlez, B. Pfitzner, M. Russo, J. Carnerero-Cano, and E. C. Lupu, "Poisoning attacks with generative adversarial nets," *arXiv*, pp. 1–15, 2019.

[8] J. Zhang, B. Chen, X. Cheng, H. T. T. Binh, and S. Yu, "PoisonGAN: Generative Poisoning Attacks against Federated Learning in Edge Computing Systems," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3310–3322, 2021, doi: 10.1109/JIOT.2020.3023126.

[9] E. Bagdasaryan and V. Shmatikov, “Blind Backdoors in Deep Learning Models,” *arXiv*, 2020.

[10] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, “How to backdoor federated learning,” *arXiv*, 2018.

[11] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, “A Survey on Distributed Machine Learning,” *ACM Computing Surveys*, vol. 53, no. 2, 2020, doi: 10.1145/3377454.

[12] Intel AI, “Federated Learning,” pp. 1–7, 19AD, [Online]. Available: https://www.intel.ai/federated-learning-for-medical-imaging/#gs.wnvtct

[13] P. Kairouz *et al.*, “Advances and open problems in federated learning,” *arXiv*, pp. 1–105, 2019, doi: 10.1561/2200000083.

[14] J. Liu *et al.*, “From Distributed Machine Learning to Federated Learning: A Survey,” 2021, [Online]. Available: http://arxiv.org/abs/2104.14362

[15] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, “A survey on security and privacy of federated learning,” *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021, doi: 10.1016/j.future.2020.10.007.

[16] A. Shafahi *et al.*, “Poison frogs! Targeted clean-label poisoning attacks on neural networks,” *Advances in Neural Information Processing Systems*, vol. 2018-Decem, no. Nips, pp. 6103–6113, 2018.

[17] T. Gu, B. Dolan-Gavitt, and S. Garg, “BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain,” 2017, [Online]. Available: http://arxiv.org/abs/1708.06733

[18] C. Fung, C. J. M. Yoon, and I. Beschastnikh, “Mitigating Sybils in Federated Learning Poisoning,” pp. 1–16, 2018, [Online]. Available: http://arxiv.org/abs/1808.04866

[19] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, “Analyzing federated learning through an adversarial lens,” *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, pp. 1012–1021, 2019.

[20] I. Goodfellow *et al.*, “Generative adversarial networks,” *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020, doi: 10.1145/3422622.

[21] A. Odena, C. Olah, and J. Shlens, “Conditional image synthesis with auxiliary classifier gans,” *34th International Conference on Machine Learning, ICML 2017*, vol. 6, pp. 4043–4055, 2017.

Cơ quan Chủ trì	Chủ nhiệm đề tài
(ký, họ và tên, đóng dấu)	*(ký, họ và tên)*
	Trần Nhật Tân