A Security-Friendly Privacy Solution for Federated Learning

Ferhat Karakoç, Leyli Karaçay, Pınar Çomak, Utku Gülen, Ramin Fuladi and Elif Ustundag Soykan

Ericsson Research, İstanbul, 34367, Turkey

Abstract

Federated learning is a privacy-aware collaborative machine learning method, but it needs other privacy-enhancing technologies to prevent data leakage from local model updates. However, privacy-enhancing technologies may not allow the usage of security mechanisms against some security attacks to model training such as poisoning and backdoor attacks. The reason is that the server that aggregates the local model updates may not be able to analyze them to detect anomalies resulting from these attacks. Solutions that satisfy both privacy and security at the same time are needed for federated learning. Another way could be introducing new privacy solutions that allow the server to execute some analysis on the local model updates without violating privacy. In this paper, we introduce a security-friendly privacy solution for federated learning, which is based on multi-hop communication to hide identities of clients but ensures that the clients in the middle points in the path between clients and the server cannot execute malicious activities such as altering model updates of other clients and sending more than one update.

Keywords

Federated learning, privacy, security attacks, poisoning attacks, multi-hop communication

1. Introduction

The enhancements in artificial intelligence (AI) and machine learning (ML) and foreseen increase in the amount of data, especially with the advent of 6G technologies, have brought advanced data-driven applications. The distributed nature of the data sources makes centralized aggregation and processing difficult due to the communication overheads. Another barrier in the full utilization of aggregated data is the privacy concerns, e.g., accessing highly sensitive data such as medical records is often prohibited. Federated learning (FL) [1] is a privacy-friendly mechanism to overcome these concerns, which generates the ML model jointly between clients and a server. Each client performs training on their local data where local training results, so-called local model updates, are transferred to the server, so the training data never leaves the clients. The server generates the global model by aggregating the local model updates. Although FL is a privacy aware solution, there are still some other privacy and security concerns

AI6G'22: First International Workshop on Artificial Intelligence in beyond 5G and 6G Wireless Networks, July 21, 2022, Padua, Italy

ferhat.karakoc@ericsson.com (F. Karakoç); leyli.karacay@ericsson.com (L. Karaçay); pinar.comak@ericsson.com (P. Çomak); utku.gulen@ericsson.com (U. Gülen); ramin.fuladi@ericsson.com (R. Fuladi); elif.ustundag.soykan@ericsson.com (E. U. Soykan)

in the FL setting. First, the local model updates sent to the server may still disclose some information about the training data of the clients to the server [2]. Second, the participants (i.e., clients) in the FL setting are able to modify the local model updates to alter the global model in their malicious aims such as performing model poisoning attacks [3, 4]. A detailed analysis of security vulnerabilities and privacy threats in FL can be found in [5]. To overcome the first concern, usage of some privacy-enhancing technologies (PETs) such as Homomorphic Encryption (HE), Secure Multi-party Computation, Differential Privacy (DP), and confidential computing have been proposed [6, 7, 8], which prevent the server from accessing the original local model updates in cleartext so that the server cannot learn any information about the training data of the clients. To address the second concern, the server can analyze the local model updates to detect any suspicious behavior by clients. Solving both the first and second concerns at the same time can be challenging. To overcome this challenge, PET-based solutions can also be utilized to allow the server to analyze local model updates without accessing them. However, the employment of these technologies may be costly, and there may be some other trust assumptions and requirements, such as the need of having non-colluding servers. FL will be vulnerable to security or privacy attacks if privacy enhanced local model analysis is not used due to the cost and/or trust assumptions. The other approach to overcome both concerns is to have privacy enhanced solutions allowing analysis of local model updates by the server without violating privacy. We call this type of solution as security-friendly privacy solution.

Related works. In the literature, several solutions to protect FL against both privacy and security attacks have been investigated. One example, MLGuard [9], introduced in 2020, where privacy of the clients is preserved by applying additive secret sharing on clients' parameter updates, and the poisoning is mitigated by the servers computing a similarity score for each user and rejecting users with lowest similarity scores (most dissimilar). Another work FLGuard [10], introduced in 2021, protects the FL process against multiple backdoor attacks, as well as provides privacy for the clients. It requires two non-colluding servers and executes a secure two-party protocol to prevent backdoor attacks and meet the privacy requirements. Another approach is presented by [11], which runs a secure aggregation protocol that is secure against malicious clients. The solution relies on the assumption that if a client or a group of clients wants to execute a backdoor attack, the malicious clients' model parameter updates diverge from the legitimate clients'. The secure aggregation protocol introduced in that work detects these divergent points so that the server recognizes the presence of a backdoor attack. In several solutions, the methods protect the FL against only privacy attacks rather than both privacy and security attacks. For example, Alberto et al. [12] propose anonymizing clients' data using multi-hop communication between the clients and the server, as an adaptation of Domingo-Ferrer et al. multi-hop protocol [13]. The server receives local model updates in plaintext, enabling the server to analyze the updates and detect security attacks. At the same time, the server will not be able to know the client to which the data belongs. However, that solution has some drawbacks that need to be solved. For example, a malicious client can alter another client's local model update and then send it to the server. Also, a malicious client can send multiple local model updates to be successful in its attack without being detected by the server. Another privacy-enhancing tool is the blind signature scheme that David Chaum introduced in 1982 [14], first employed for untraceable payment applications. This scheme is a type of digital signature in which the content of the private data is disguised before being signed

by a signature issuer (server). Therefore, the issuer cannot obtain any information regarding the content of the data. Thus the confidentiality of the data is guaranteed. One enhancement in the blind signature is the introduction of partially blind signatures [15, 16] where the signature issuer (server) can input additional common information, agreed by the client and issuer, into the data before signing the data of the client. Partially blind signature schemes are widely used in electronic cash systems and electronic voting. Also, some other applications of partially blind signatures have also been introduced in the literature. For example, Gong et al. used partially blind signatures in smart grid applications [17]. Buccafurri et al. utilized partially blind signature in privacy-preserved analysis of social media "likes" [18]. Blind signatures found an application area in privacy preserving communication of VANETs in Fan et al. study [19]. Li et al. proposed a partially blind signature based technique for privacy-preserving of participatory sensing [20].

Our contribution. We improve the multi-hop communication solution proposed by Alberto et al. [12] using blind signatures to solve the problems of possible malicious behaviors of clients. To be able to detect unauthorized modifications to the local model updates in the multi-hop communication, usage of signatures for the local model updates by the owner client is proposed. To address the other malicious behavior of sending multiple local model updates, we utilize partially blind signatures. The server provides signed public keys, to be used only once, to the clients blindly, and the clients will be able to use the corresponding private keys to sign local model updates. The main contributions of this paper are as follows:

- We propose a privacy attacks mitigation technique that preserves the privacy of clients by preventing directly sending local model updates to server using multi-hop communication.
- The proposed method enables the server to identify some possible malicious behavior of the clients who are willing to alter the model updates or send multiple local model updates.
- The proposed method enables the server to prevent malicious clients' local model updates from disturbing the global model.

The rest of the paper is organized as follows. Section 2 provides the preliminaries relating to the tools utilized in this paper. Section 3 discusses the proposed protocol. Finally, Section 4 concludes the paper.

2. Preliminaries

2.1. Notation

Throughout the paper, the following notation is used. (pk_i, sk_i) denotes the public and private key pair of the i-th client. c is to represent the commuting function used in the blind signature. s' is the blind signature operation of the server using its private key. c' and s are the inverse of c and s', respectively. n denotes the number of clients in the FL setting. m is the tolerable number of lost local model updates in each FL round. $\mathcal{G}_{\mathcal{FED}}$ is the global model and δ_i^j denotes the local model update for user i at round j.

2.2. Federated Learning

FL is a privacy-friendly collaborative ML technique where model training is distributed among multiple clients. In this learning scenario, a server initializes a global model $\mathcal{G}_{\mathcal{FED}}$ and sends it to n clients where all clients wish to train a machine learning model using their respective data D_i for $i = 1, \ldots, n$. Each FL process has several rounds such that at each round t (or iteration) of the federated learning, each client trains on its local data D_i and share only local model updates δ_i^t with the server for iterative aggregation until a convergent global model $\mathcal{G}_{\mathcal{FED}}$ is reached on the server. The accuracy of the $\mathcal{G}_{\mathcal{FED}}$ should be very close to the conventional method where all data from clients are put together to train a model [21].

2.3. Blind Signatures

Blind signatures, introduced by David Chaum [14], allow one party to have a signature for its private data from a signature issuer without leaking any information about the data. Chaum proposed to use this approach for untraceable payments. The blind signature protocol works as follows:

1. The client computes $c(x)$ where x is the input for the signature and c is the commuting function that is only known by the client. Then the client sends $c(x)$ to the server.
2. The server computes the signature operation to have $s'(c(x))$ where s' is the signature operation including the private key of the server. Then server send $s'(c(x))$ to the client.
3. The client recovers $s'(x)$ which is the server's signature on x by computing $c'(s'(c(x)))$ where c' is the inverse of c and only known by the client.

The usage of this blind signature concept for an untraceable payment works as follows:

1. The payer chooses a random x, computes $c(x)$, and sends the result to the bank.
2. The bank signs it ($s'(c(x))$), debits the payer's account, and sends the signature result to the payer.
3. The payer recovers the signature $c'(s'(c(x))) = s'(x)$ and verifies the signature using s where s is the signature verification including the public key of the Bank.
4. The payer sends $s'(x)$ to a payee.
5. The payee sends $s'(x)$ to the bank.
6. The bank verifies the signature and if the verification is successful, then checks whether x has been used before. Since the bank blindly signs x in step 2, the bank will not be able to match the payer and payee; so, privacy will be satisfied. If x has been used before, the bank refuses the payment; otherwise, the bank credits the account of the payee.

Partially blind signatures were introduced in [15, 16] where the signature issuer (server) can input additional common information such as date or other agreed information into the data before signing the data of the client. Various implementations of partially blind signature have been proposed by using different cryptographic primitives such as RSA, ECDSA, and bilinear pairings [22, 23, 24].

3. Our protocol

We build our protocol on top of the solution proposed by Alberto et al. [12] where the main idea is to anonymize the local model updates sent to the server. This privacy approach allows the server to analyze the incoming data to detect any poison or backdoor attack attempts. One of the drawbacks of the solution is that since there is no data source authentication, malicious clients in the path between legitimate clients and server may utilize this anonymization technique to execute some attacks by modifying other clients' local model updates before forwarding them or by sending multiple local model updates in one round of FL to alter the global model in their pursuit without being detected by the server. To prevent the clients from sending more than one local model update, we utilize cryptographic signatures for the source authentication. Not to break anonymity, we need to ensure that the server should not be able to learn the identities of the clients from the signatures. To address these two requirements, cryptographic signature and to hide the identities, we use blind signatures. The example interactions between server and clients are illustrated in Figure 1.

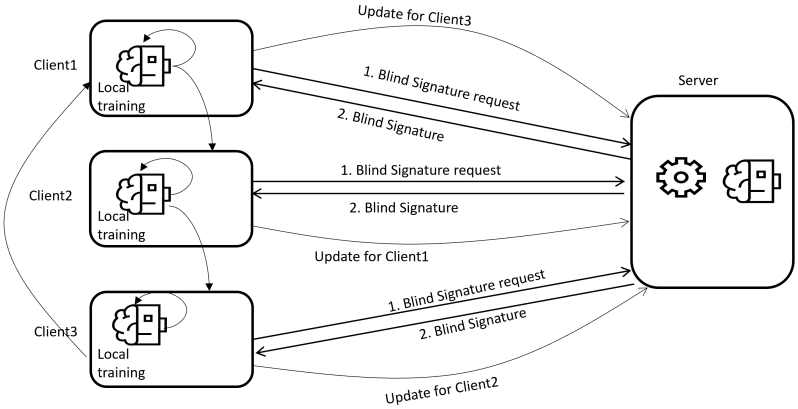

Figure 1: Example interactions between server and clients

In our trust model, we consider the server as a malicious party in terms of privacy, who may want to learn information about training data of clients. We also consider the clients as malicious parties who may try to alter the local model updates received from other clients, send multiple local model updates for one round of FL, or may want to learn information about the training data of other clients. The general idea in our protocol is that the server signs public keys for the clients blindly, so that the server should not learn the public keys of the clients while signing them. Also, to bind the public keys to a specific FL round, the server includes the current FL round number in the signature. To be able to insert this common information into the signature, we utilize partially blind signatures. For that purpose, the server signs the public keys sent by the clients blindly. This operation is repeated for each round of FL. With this blind public key signing operation, the server ensures that it provides only one public key per client for each FL round. Before sending the local model update, each client signs its own update, and then the server validates the signature on the public keys and the signature on the local model updates. The server also stores the used public keys in a table to ensure that the

public key is used only once. Thus, this solution allows the server to ensure that the clients in the setting have only one public key per FL round and also ensure that the clients can use the public keys to sign only one local model update. It is important for the clients to check that they have the same public key of the server where the public key is the corresponding private key used in the blind signature. Otherwise, the server may try to violate the privacy of clients. We remark that the communication between the clients and server needs to be secured using confidentiality, integrity and authentication methods to ensure that the clients and the server are legitimate entities and the model is not revealed to anyone who is not in the FL setting such as man-in-the-middle attackers. The steps of our protocol, which are executed in each FL round, are as follows.

1. Each client generates a public-private key pair (pk_i and sk_i for the i-th client)
2. Each client runs a partially blind signature protocol with the server to have blindly signed public key (pk_i). Here the common information included in the partially blind signature is the current FL round number. $s'(pk_i, r_j)$ denotes the blindly signed public key where r_j is the current FL round number.
3. Then each client performs local training to obtain the local model update.
4. Each client signs the local model update using sk_i. Also, selects a random hop-counter and sends pk_i, r_j, $s'(pk_i, r_j)$, the local model update and its signature, and the hop-counter to a randomly selected forwardee.
5. Each client who receives a packet decreases the hop-counter in the received packet. The forwardee sends the received sk_i, r_j, $s'(pk_i, r_j)$, the signature of the local model update, the local model update, and the hop counter to the server or to another client.
6. The server executes the following steps when it receives a packet.
   a) Checks whether sk_i is used before for the r_j-th round.
   b) Verifies $s'(pk_i, r_j)$.
   c) Verifies the signature on the local model update using the received public key pk_i.
7. The server proceeds to the aggregation operation as long as it receives at least $n - m$ data packets that pass the checks in step 6. Thus, it is ensured that the FL is not disrupted if a client does not forward the model updates for malicious reasons, i.e., denial-of-service by malicious clients is somewhat reduced. Here, the parameter m can be calculated considering the probability of receiving m packets by the malicious client, using the number of clients and hop counts. Note that the server can also execute some analysis on the received local model updates to detect any poisoning attacks.

Security arguments. Our protocol aims to achieve the security requirements listed below. We also give sketched security arguments to show that our protocol meets these requirements.

1. *The server should not be able to learn the owner of the local model update.* Since the local model updates are sent to the server in a multi-hop communication, the server will not be able to learn the source client. The other way for the server to cheat is that the server can behave maliciously in the partially blind signature operation. Since we assume that the partially blind signature protocol is secure, the only way for the server is to use different keys for the blind signature operation to detect which public key belongs to which client,

but the clients will detect this attempt because they check that the public key of the server is same for all clients.

2. *The server will be able to make some analysis on the local model updates to detect security attacks to the model training.* Since the local model updates are sent in cleartext to the server, the server will be able to execute some protection mechanisms against security attacks by analyzing received local model updates.
3. *Forwardee clients should not be able to learn the owner of the local model update received from a client.* Since the owner of the local model update includes a random hop counter to the packet sent to the forwardee client, the forwardee client will not be able to know whether the sender is the owner of the local model update.
4. *Forwardee clients should not be able to alter the local model update received from a client.* This is achieved with the usage of signatures on local model updates.
5. *Clients should not be able to send multiple local model updates to the server.* This is ensured as the server provides only one signed public key to each client for each FL round and checks whether the received public key has been used before.
6. *No one except the clients in the setting will be able to send local model updates to the server.* This is guaranteed since the server accepts local model updates from the clients whose public keys are blindly signed by the server. The server checks the server signature on the public key used to sign the local model update.

Performance considerations. We discuss the overheads of our protocol. The clients need different key pairs for each federated learning round. The cost of generating key pairs can be eliminated by generating them offline. The other additional step is the blind signature and model signature operations. Thus, the server needs to compute n blind signatures in step 2, and in step 6 needs to compute n blind signature verifications along with n normal signature verifications. For the clients, the computational overhead is the computation of one blind signature-related operation in Step 2 and the local model signature operation in Step 4. For the communication, each client needs to connect to the server to have blindly signed public keys. Also, the signed local model updates need to be sent hop-by-hop instead of directly sent to the server, which may cause some delay in the FL process, but this doesn't bring considerable computation overhead to the forwardee clients.

4. Conclusion

The adoption of collaborative ML/AI, especially FL, will increase in future generation networks to address distributed nature of data analytics and computations stemming from emerging use cases such as internet of senses and extended reality. It is important to provide trustworthy methodologies for FL. With this motivation, we propose a new method to improve security and privacy in FL. Our method is based on a new approach that uses partially blind signatures to resolve residual privacy and security issues of the multi-hop communication approach for anonymization of clients participating in the FL training, suggested in [12]. Our approach mitigates the possible malicious behaviors of clients. We consider two types of malicious

behavior (i) unauthorized modifications to the local model update during transmission to the server (ii) sending multiple local model updates to the server. We address the first one by introducing signatures for the local model updates by the owner client and the second one by using partially blind signatures where the server provides one usage signed public key to the clients blindly, and the clients use the public keys to sign local model updates only once. To the best of our knowledge, our solution is the first solution that uses partially blind signatures in the federated learning setting.

Acknowledgments

This work was supported by The Scientific and Technological Research Council of Turkey (TUBITAK) through the 1515 Frontier Research and Development Laboratories Support Program under Project 5169902 and has been partly funded by the European Commission through the H2020 project Hexa-X (Grant Agreement no. 101015956).

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: AISTATS, 2017.

[2] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., Advances and open problems in federated learning, Foundations and Trends® in Machine Learning 14 (2021) 1–210.

[3] X. Zhou, M. Xu, Y. Wu, N. Zheng, Deep model poisoning attack on federated learning, Future Internet 13 (2021) 73.

[4] L. Lyu, H. Yu, J. Zhao, Q. Yang, Threats to federated learning, in: Federated Learning, Springer, 2020, pp. 3–16.

[5] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, G. Srivastava, A survey on security and privacy of federated learning, Future Generation Computer Systems 115 (2021) 619–640.

[6] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, K. Seth, Practical secure aggregation for privacy-preserving machine learning, in: proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 1175–1191.

[7] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, H. V. Poor, Federated learning with differential privacy: Algorithms and performance analysis, IEEE Transactions on Information Forensics and Security 15 (2020) 3454–3469.

[8] J. Park, H. Lim, Privacy-preserving federated learning using homomorphic encryption, Applied Sciences 12 (2022) 734.

[9] Y. Khazbak, T. Tan, G. Cao, Mlguard: Mitigating poisoning attacks in privacy preserving distributed collaborative learning, in: 2020 29th International Conference on Computer Communications and Networks (ICCCN), IEEE, 2020, pp. 1–9.

[10] T. D. Nguyen, P. Rieger, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen,

A. Mirhoseini, A. Sadeghi, T. Schneider, S. Zeitouni, FLGUARD: secure and private federated learning, IACR Cryptol. ePrint Arch. 2021 (2021) 25.

[11] F. Karakoç, M. Önen, Z. Bilgin, Secure aggregation against malicious users, in: J. Lobo, R. D. Pietro, O. Chowdhury, H. Hu (Eds.), SACMAT '21: The 26th ACM Symposium on Access Control Models and Technologies, Virtual Event, Spain, June 16-18, 2021, ACM, 2021, pp. 115–124.

[12] A. Blanco-Justicia, J. Domingo-Ferrer, S. Martínez, D. Sánchez, A. Flanagan, K. E. Tan, Achieving security and privacy in federated learning systems: Survey, research challenges and future directions, Eng. Appl. Artif. Intell. 106 (2021) 104468.

[13] J. Domingo-Ferrer, S. Martínez, D. Sánchez, J. Soria-Comas, Co-utility: Self-enforcing protocols for the mutual benefit of participants, Eng. Appl. Artif. Intell. 59 (2017) 148–158.

[14] D. Chaum, Blind signatures for untraceable payments, in: D. Chaum, R. L. Rivest, A. T. Sherman (Eds.), Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982, Plenum Press, New York, 1982, pp. 199–203.

[15] M. Abe, E. Fujisaki, How to date blind signatures, in: K. Kim, T. Matsumoto (Eds.), Advances in Cryptology - ASIACRYPT '96, International Conference on the Theory and Applications of Cryptology and Information Security, Kyongju, Korea, November 3-7, 1996, Proceedings, volume 1163 of *Lecture Notes in Computer Science*, Springer, 1996, pp. 244–251.

[16] M. Abe, T. Okamoto, Provably secure partially blind signatures, in: M. Bellare (Ed.), Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings, volume 1880 of *Lecture Notes in Computer Science*, Springer, 2000, pp. 271–286.

[17] Y. Gong, Y. Cai, Y. Guo, Y. Fang, A privacy-preserving scheme for incentive-based demand response in the smart grid, IEEE Trans. Smart Grid 7 (2016) 1304–1313.

[18] F. Buccafurri, L. Fotia, G. Lax, V. Saraswat, Analysis-preserving protection of user privacy against information leakage of social-network likes, Inf. Sci. 328 (2016) 340–358.

[19] C. Fan, W. Sun, S. Huang, W. Juang, J. Huang, Strongly privacy-preserving communication protocol for vanets, in: Ninth Asia Joint Conference on Information Security, AsiaJCIS 2014, Wuhan, China, September 3-5, 2014, IEEE Computer Society, 2014, pp. 119–126.

[20] Q. Li, G. Cao, Privacy-preserving participatory sensing, IEEE Commun. Mag. 53 (2015) 68–74.

[21] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: Concept and applications, ACM Trans. Intell. Syst. Technol. 10 (2019). doi:10.1145/3298981.

[22] H. Chien, J. Jan, Y. Tseng, Rsa-based partially blind signature with low computation, in: Eigth International Conference on Parallel and Distributed Systems, ICPADS 2001, KyongJu City, Korea, June 26-29, 2001, IEEE Computer Society, 2001, pp. 385–389.

[23] H. Huang, Z. Liu, R. Tso, Partially blind ECDSA scheme and its application to bitcoin, in: IEEE Conference on Dependable and Secure Computing, DSC 2021, Aizuwakamatsu, Japan, January 30 - February 2, 2021, IEEE, 2021, pp. 1–8.

[24] A. Koide, R. Tso, E. Okamoto, Convertible undeniable partially blind signature from bilinear pairings, in: C. Xu, M. Guo (Eds.), 2008 IEEE/IPIP International Conference on Embedded and Ubiquitous Computing (EUC 2008), Shanghai, China, December 17-20, 2008, Volume II: Workshops, IEEE Computer Society, 2008, pp. 77–82.