Comments on “VERSA: Verifiable Secure Aggregation for Cross-Device Federated Learning”

Yanxin Xu, Hua Zhang, *Member, IEEE,* Shaohua Zhao, Xin Zhang, Wenmin Li, Fei Gao, and Kaixuan Li

Abstract—Recently, in IEEE Transactions on Dependable and Secure Computing (TDSC), the VERSA scheme proposed by Hahn et al. uses a double aggregation method for verifying the correctness of results returned from the server. The authors proposed that the correctness of the model aggregation can be verified with lower verification overhead by utilizing only a lightweight pseudorandom generator. To support verifiability of results returned from the server, a method of sharing a pair of vectors (a, b) by all clients is proposed, which is one of the most important work in VERSA. Unfortunately, in this paper, we show that the method is incorrect, which leads clients to consistently conclude that the aggregated results are incorrect. Furthermore, the model training process in federated learning is forced to abort. Finally, we demonstrate our view through theory analysis and instantiation verification.

Index Terms—Federated learning, incorrectness, verifiable aggregation, VERSA.

1 Introduction

The server in federated learning(FL) can easily become a single-point of failure or is driven by certain illegal interests. For violating the privacy of clients or launching other attacks on them, the server may return incorrect aggregated results to clients. Therefore, it is critical to ensure the correctness of the aggregated results.

To address this issue, Hahn et al. [1] proposed the VERSA, which utilizes only a lightweight pseudorandom generator to verify the correctness of the aggregated results, with lower communication overhead and computation overhead. Specifically, in VERSA, all clients utilize the same shared pair of vectors to transform their local gradients respectively, and the aggregated result still satisfies the same transformation relationship.

Unfortunately, we find that the shared vectors generated in this scheme between different clients are different. This causes aggregated results do not satisfy the same transformation relationship as local gradients being transformed. Thereby, all clients conclude that the server does not implement the aggregation protocol faithfully. Then, they reject the global model, and the model training process in FL is forced to abort.

2 Review

In this paper, we focus on the process of verifying the correctness of aggregated results returned from the server. We first introduce some notations used in the VERSA scheme [1]. Let $s_{u,v} \leftarrow$ **KA.agree** (sk_u, pk_v) be a key agreement (KA) protocol. This algorithm takes private key sk_u of user u and public key pk_v of user v as input, and outputs a shared key $s_{u,v}$. Let $\mathbb{Z}_R^d \leftarrow$ **PRG** $\left(\{0,1\}^*\right)$ be a pseudorandom generator (PRG). The PRG expands an input value into a d-dimensional vector of modulo R, where R is a large integer.

Key Generation: Let U is a set of clients participating in FL. Each client $u \in U$ first runs KA protocol locally with its private key and all public keys of other clients to obtain a set of secret values $\{s_{u,v}\}_{v\in U\setminus u}$. Next, client u computes another secret value $\alpha \leftarrow \sum_{v\in U\setminus u} s_{u,v} \pmod R$. Then, the PRG is utilized by client u to expand α into two random vectors $a = PRG(\alpha \parallel 0)$ and $b = PRG(\alpha \parallel 1)$. The same pair of vectors (a, b) can be obtained by all clients, and the two vectors are secret vectors hidden from the server view.

Masking Input: Each client $u \in U$ submits y_u and $\bar{y_u}$ to the server. Specifically, y_u is the ciphertext of the local gradient vector x_u encrypted with a double masking operation. Similarly, $\bar{y_u}$ is the ciphertext of the model verification code $F(x_u)$, where $F(x_u) = a \circ x_u + b$ and operation $\circ$ is the Hadamard product.

Server Aggregation: The server receives a set of ciphertexts $\{y_u, \bar{y_u}\}_{u\in U}$. Next, it performs the aggregation operation twice to obtain the aggregated gradient $z = \sum_{u\in U} y_u = \sum_{u\in U} x_u$ and the aggregated model verification code $\bar{z} = \sum_{u\in U} \bar{y_u} = \sum_{u\in U} F(x_u)$. The pair of vectors $(z, \bar{z})$ is broadcast to all clients.

Clients Verification: Each client $u \in U$ receives a pair of vectors $(z, \bar{z})$ from the server and computes a value $z' = a \circ z + |U| \cdot b \pmod R$. Eventually, clients verify z by checking whether the following condition holds: $\bar{z} \overset{?}{=} z'$. If the condition holds, they accept the aggregated result z and run local training. Otherwise, they reject the aggregated result z.

- *Authors are all with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China.*
E-mail: yanxinxu@bupt.edu.cn, zhanghua_288@bupt.edu.cn, zhaoshaohua@bupt.edu.cn, zxby233@bupt.edu.cn, liwenmin02@outlook.com, gaof@bupt.edu.cn, lkx1118@bupt.edu.cn
- *H. Zhang is the corresponding author.*
E-mail: zhanghua_288@bupt.edu.cn

Manuscript received April 19, 2005; revised August 26, 2015.

3 Incorrectness of VERSA

One of the most important work in VERSA is enabling all clients to share the same pair of vectors (a, b). According to the authors, this issue is addressed by using a secret expansion through the PRG, as described in the key generation phase. Unfortunately, we find that all clients in U cannot obtain the same set of secret values $\{s_{u,v}\}_{v\in U\setminus u}$. Eventually, all clients compute different secret values α and cannot obtain the same pair of vectors (a, b).

3.1 Theory Analysis

We assume that the PRG is collision-resistant. For any two different clients u_i and u_j, their private keys are different, i.e., $sk_{u_i} \neq sk_{u_j}$. In addition, pk_{u_i} and pk_{u_j} are their public keys respectively.

According to the protocol in VERSA, client u_i first takes its private key and public keys of all other clients as the input of the KA protocol. u_i obtains a set of keys $\{s_{u_i,v}\}_{v\in U\setminus u_i}$, and computes another secret value

$$\alpha_i = s_{u_i,u_j} + \textstyle\sum_{v\in U\setminus(u_i\cup u_j)} s_{u_i,v} \pmod R,$$

where $s_{u_i,v} \leftarrow \textbf{KA.agree}\left(sk_{u_i}, pk_v\right)$.

Similarly, client u_j obtains a set of keys $\{s_{u_j,v}\}_{v\in U\setminus u_j}$, and computes another secret value

$$\alpha_j = s_{u_j,u_i} + \textstyle\sum_{v\in U\setminus(u_i\cup u_j)} s_{u_j,v} \pmod R,$$

where $s_{u_j,v} \leftarrow \textbf{KA.agree}\left(sk_{u_j}, pk_v\right)$ and $s_{u_i,u_j} = s_{u_j,u_i}$.

Next, the PRG is utilized by client u_i to expand α_i into two random vectors $a_i = \textbf{PRG}\left(\alpha_i \parallel 0\right)$ and $b_i = \textbf{PRG}\left(\alpha_i \parallel 1\right)$. Similarly, client u_j obtains $a_j = \textbf{PRG}\left(\alpha_j \parallel 0\right)$ and $b_j = \textbf{PRG}\left(\alpha_j \parallel 1\right)$.

If $a_i = a_j$ and $b_i = b_j$, clients u_i and u_j can obtain the same pair of vectors (a, b) by utilizing the method in VERSA. Since the PRG is collision-resistant, we obtain $\alpha_i \parallel 0 = \alpha_j \parallel 0$ and $\alpha_i \parallel 1 = \alpha_j \parallel 1$. Then $\alpha_i = \alpha_j$ is obtained.

Due to $\alpha_i = s_{u_i,u_j} + \sum_{v\in U\setminus(u_i\cup u_j)} s_{u_i,v} \pmod R$, $\alpha_j = s_{u_j,u_i} + \sum_{v\in U\setminus(u_i\cup u_j)} s_{u_j,v} \pmod R$ and $s_{u_i,u_j} = s_{u_j,u_i}$, it can be inferred that if clients u_i and u_j obtain the same pair of vectors (a, b), the condition

$$\textstyle\sum_{v\in U\setminus(u_i\cup u_j)} s_{u_i,v} \pmod R = \textstyle\sum_{v\in U\setminus(u_i\cup u_j)} s_{u_j,v} \pmod R \quad (1)$$

must be held.

However, because of $s_{u_i,v} \leftarrow \textbf{KA.agree}\left(sk_{u_i}, pk_v\right)$, $s_{u_j,v} \leftarrow \textbf{KA.agree}\left(sk_{u_j}, pk_v\right)$ and $sk_{u_i} \neq sk_{u_j}$, we obtain $s_{u_i,v} \neq s_{u_j,v}$ $(v \in U \setminus (u_i \cup u_j))$. Then the conclusion

$$\textstyle\sum_{v\in U\setminus(u_i\cup u_j)} s_{u_i,v} \pmod R \neq \textstyle\sum_{v\in U\setminus(u_i\cup u_j)} s_{u_j,v} \pmod R$$

is obtained, which contradicts the previous conclusion, i.e., formula (1). Therefore, in this condition there exist two different clients that cannot obtain the same pair of vectors (a, b).

By proofing with reductio ad absurdum, we infer that by utilizing the method in VERSA, the same pair of vectors (a, b) cannot be shared by all clients in U. Particularly, when clients verify the aggregated results by checking whether the following condition holds: $\bar{z} \overset{?}{=} z'$, they conclude that the condition does not hold. Thereby, all clients conclude that the server does not implement the aggregation protocol faithfully and reject the global model.

3.2 Instantiation Verification

In the evaluation of VERSA, the authors used the elliptic curve Diffie-Hellman (ECDH) agreement [2] and SHA-256 [3] to implement the KA and PRG respectively. Therefore, we utilize the same algorithms to further verify the incorrectness of the scheme.

Each client $u \in U$ chooses its private key n_u independently, and its public key P_u is computed, where n_u is an integer, $P_u = n_u G$, and G is the generator of an Abel group.

Client u_i first takes its private key and public keys of all other clients as the input of the ECDH protocol. u_i obtains a set of keys $\{s_{u_i,v} = n_{u_i} P_v\}_{v\in U\setminus u_i}$, and computes another secret value

$$\alpha_i = n_{u_i} n_{u_j} G + n_{u_i} \textstyle\sum_{v\in U\setminus(u_i\cup u_j)} n_v G \pmod R.$$

Similarly, client u_j obtains a set $\{s_{u_j,v} = n_{u_j} P_v\}_{v\in U\setminus u_j}$, and computes another secret value

$$\alpha_j = n_{u_j} n_{u_i} G + n_{u_j} \textstyle\sum_{v\in U\setminus(u_i\cup u_j)} n_v G \pmod R.$$

Since n_{u_i} and n_{u_j} are private values independently selected by clients u_i and u_j respectively, we generally think that $n_{u_i} \neq n_{u_j} \pmod R$. Therefore, if $i \neq j$, then $\alpha_i \neq \alpha_j$.

Next, the algorithm SHA-256 is utilized by client u_i to expand α_i into two random vectors $a_i = \textbf{SHA}_{\textbf{256}}\left(\alpha_i \parallel 0\right)$ and $b_i = \textbf{SHA}_{\textbf{256}}\left(\alpha_i \parallel 1\right)$. Similarly, client u_j obtains $a_j = \textbf{SHA}_{\textbf{256}}\left(\alpha_j \parallel 0\right)$ and $b_j = \textbf{SHA}_{\textbf{256}}\left(\alpha_j \parallel 1\right)$.

If $i \neq j$, then $\alpha_i \neq \alpha_j$. Since the algorithm SHA-256 is collision-resistant, $a_i \neq a_j$ and $b_i \neq b_j$ are obtained. Therefore, for any two different clients u_i and u_j, they cannot obtain the same pair of vectors (a, b) by utilizing the method in VERSA.

With instantiation verification, we show that by utilizing the method in VERSA, the same pair of vectors (a, b) cannot be shared by all clients.

4 Conclusion

In this paper, we studied the VERSA scheme in [1] and showed that the method of implementing the same pair of vectors shared by all clients in this scheme is incorrect. When clients verify the aggregated results returned from the server, they consistently conclude that the results are incorrect. Moreover, they always think that the server does not implement the aggregation protocol faithfully. Thus, the model training process in FL is forced to abort.

Acknowledgments

This work is supported by NSFC (Grant No. 62072051, 61976024, 61972048).

References

[1] C. Hahn, H. Kim, M. Kim, and J. Hur, "Versa: Verifiable secure aggregation for cross-device federated learning," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[2] N. Koblitz, A. Menezes, and S. Vanstone, "The state of elliptic curve cryptography," *Designs, codes and cryptography*, vol. 19, no. 2, pp. 173–193, 2000.

[3] S. Indesteege, F. Mendel, B. Preneel, and C. Rechberger, "Collisions and other non-random properties for step-reduced sha-256," in *International Workshop on Selected Areas in Cryptography*. Springer, 2008, pp. 276–293.