A Note on "Privacy Preserving n-Party Scalar Product Protocol"

Lihua Liu

Abstract. We show that the scalar product protocol [IEEE Trans. Parallel Distrib. Syst. 2023, 1060-1066] is insecure against semi-honest server attack, not as claimed. Besides, its complexity increases exponentially with the number n, which cannot be put into practice.

Keywords: privacy preserving scalar product, semi-honest server attack, diagonal matrix, trace map.

1 Introduction

In 2002, Du and Zhan [1] designed a privacy preserving two-party scalar product protocol. Recently, Daalen et al. [2] have generalized it to a general n-party protocol. In this note, we show that the Daalen et al.'s protocol is insecure against semi-honest server attack. Besides, the protocol cannot be practically implemented because of its exponential complexity.

2 The Du-Zhan two-party protocol

Alice and Bob want to calculate the scalar product of their private vectors A and B, both of the same size m. Merlin is a semi-honest server, who generates two random vectors R_a, R_b of size m and two scalars r_a and r_b such that

$$r_a + r_b = R_a \cdot R_b \tag{1}$$

Then securely send (R_a, r_a) to Alice, and (R_b, r_b) to Bob. The protocol can be depicted as below (Table 1).

Clearly, we have

$$\begin{aligned} & v_1 + v_2 = u - R_a \cdot \hat{B} + r_a + v_2 \\ & = \hat{A} \cdot B + r_b - v_2 - R_a \cdot \hat{B} + r_a + v_2 \\ & = (A + R_a) \cdot B - R_a \cdot (B + R_b) + r_a + r_b \\ & = A \cdot B - R_a \cdot R_b + r_a + r_b = A \cdot B \end{aligned}$$

Notice that the Du-Zhan protocol has two shortcomings. The first is the presence of an honest convener. In the protocol, only Bob can know the result $v_1 + v_2$. If Alice wants to know the result, it must introduce other mechanism enabling Bob to honestly and securely transfer the nonce v_2 to Alice.

L. Liu is with Department of Mathematics, Shanghai Maritime University, Shanghai, China. Email: liulh@shmtu.edu.cn

Table 1: The Du-Zhan two-party protocol

Alice: (R_a, r_a)		Bob: (R_b, r_b)
Input the private A. Compute the vector $\hat{A} = A + R_a$.	$\xrightarrow{\hat{A}}$	
	$\xleftarrow{\hat{B},u}$	Input the private B. Pick a nonce v_2. Compute $\hat{B} = B + R_b$, $u = \hat{A} \cdot B + r_b - v_2$.
Compute the scalar $v_1 = u - R_a \cdot \hat{B} + r_a$.	$\xrightarrow{v_1}$	
		Output $v_1 + v_2$.

The second is that insecurity against semi-honest server attack. A semi-honest party is a party which executes its part in the protocol accurately, but may try to learn as much as it can from the messages it receives in the process. In the Du-Zhan protocol, once the server captured $\hat{A}, \hat{B}$ via the open channels, it can retrieve the private vectors A, B using R_a, R_b.

3 The Daalen et al.'s three-party protocol

3.1 Review of the Daalen et al.'s protocol

Table 2: The Daalen et al.'s three-party scalar product protocol

Alice: $(\boldsymbol{R}_a, r_a)$	Bob: $(\boldsymbol{R}_b, r_b)$	Claire: $(\boldsymbol{R}_c, r_c)$
Input the private A. Convert it into $\boldsymbol{A}$. Compute $\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{R}_a$. Perform two-party scalar product protocol with Merlin who knows $\boldsymbol{R}_b, \boldsymbol{R}_c$ to obtain $k_a = \phi(\boldsymbol{A}\boldsymbol{M}_a)$, where $\boldsymbol{M}_a = \boldsymbol{R}_b\boldsymbol{R}_c$. Broadcast $\hat{\boldsymbol{A}}$.	Input the private B. Convert it into $\boldsymbol{B}$. Compute $\hat{\boldsymbol{B}} = \boldsymbol{B} + \boldsymbol{R}_b$. Perform two-party scalar product protocol with Merlin who knows $\boldsymbol{R}_a, \boldsymbol{R}_c$ to obtain $k_b = \phi(\boldsymbol{B}\boldsymbol{M}_b)$, where $\boldsymbol{M}_b = \boldsymbol{R}_a\boldsymbol{R}_c$. Broadcast $\hat{\boldsymbol{B}}$.	Input the private C. Convert it into $\boldsymbol{C}$. Compute $\hat{\boldsymbol{C}} = \boldsymbol{C} + \boldsymbol{R}_c$. Perform two-party scalar product protocol with Merlin who knows $\boldsymbol{R}_a, \boldsymbol{R}_b$ to obtain $k_c = \phi(\boldsymbol{C}\boldsymbol{M}_c)$, where $\boldsymbol{M}_c = \boldsymbol{R}_a\boldsymbol{R}_b$. Broadcast $\hat{\boldsymbol{C}}$.
	Pick a nonce v_2, compute $u_1 = \phi(\boldsymbol{B}\hat{\boldsymbol{A}}\hat{\boldsymbol{C}}) + 2r_b - v_2$. $\xleftarrow{u_1,\ k_b}$	
Compute $u_2 = u_1 - \phi(\boldsymbol{R}_a\hat{\boldsymbol{B}}\hat{\boldsymbol{C}}) + 2r_a$.	$\xrightarrow{u_2,\ k_a,\ k_b}$	$u_3 = u_2 - \phi(\boldsymbol{R}_c\hat{\boldsymbol{A}}\hat{\boldsymbol{B}}) + 2r_c + k_a + k_b + k_c$.
	Output $u_3 + v_2$.	$\xleftarrow{u_3}$

In the Daalen et al.'s three-party protocol [2], we use lowercase letters to denote scalars, uppercase for vectors and uppercase with a bold face for matrices. It needs to convert a vector into its corresponding diagonal matrix. Let ϕ be the trace map of a matrix. The protocol can be rephrased and depicted as below (Table 2).

The server Merlin generates three random diagonal matrices $\boldsymbol{R}_a$, $\boldsymbol{R}_b$, $\boldsymbol{R}_c$ and three scalars r_a, r_b, r_c such that

$$r_a + r_b + r_c = \phi(\boldsymbol{R}_a\boldsymbol{R}_b\boldsymbol{R}_c) \tag{2}$$

Send $\{\boldsymbol{R}_a, r_a\}$ to Alice, $\{\boldsymbol{R}_b, r_b\}$ to Bob and $\{\boldsymbol{R}_c, r_c\}$ to Claire.

Its correctness is due to that

$$\begin{aligned}
u_3 &= u_2 - \phi(\boldsymbol{R}_c\hat{\boldsymbol{A}}\hat{\boldsymbol{B}}) + 2r_c + k_a + k_b + k_c \\
&= u_1 - \phi(\boldsymbol{R}_a\hat{\boldsymbol{B}}\hat{\boldsymbol{C}}) + 2r_a - \phi(\boldsymbol{R}_c\hat{\boldsymbol{A}}\hat{\boldsymbol{B}}) \\
&\quad + 2r_c + k_a + k_b + k_c \\
&= \phi(\hat{\boldsymbol{A}}\hat{\boldsymbol{C}}\boldsymbol{B}) + 2r_b - v_2 - \phi(\boldsymbol{R}_a\hat{\boldsymbol{B}}\hat{\boldsymbol{C}}) \\
&\quad + 2r_a - \phi(\boldsymbol{R}_c\hat{\boldsymbol{A}}\hat{\boldsymbol{B}}) + 2r_c + k_a + k_b + k_c \\
&= \phi((\boldsymbol{A} + \boldsymbol{R}_a)(\boldsymbol{C} + \boldsymbol{R}_c)\boldsymbol{B}) - \phi(\boldsymbol{R}_a(\boldsymbol{B} + \boldsymbol{R}_b)(\boldsymbol{C} + \boldsymbol{R}_c)) \\
&\quad - \phi(\boldsymbol{R}_c(\boldsymbol{A} + \boldsymbol{R}_a)(\boldsymbol{B} + \boldsymbol{R}_b)) \\
&\quad + 2(r_a + r_b + r_c) - v_2 + k_a + k_b + k_c \\
&= \phi(\boldsymbol{ABC}) - \phi(\boldsymbol{A}\boldsymbol{R}_b\boldsymbol{R}_c) - \phi(\boldsymbol{B}\boldsymbol{R}_a\boldsymbol{R}_c) - \phi(\boldsymbol{C}\boldsymbol{R}_a\boldsymbol{R}_b) \\
&\quad - v_2 + k_a + k_b + k_c = \phi(\boldsymbol{ABC}) - v_2
\end{aligned}$$

The three-party protocol can be generalized to an n-party protocol, but which should recursively perform plenty of $(n-1)$-party protocols, $(n-2)$-party protocols, ..., and 2-party protocols. We refer to the original description (page 1062, [2]).

3.2 Insecure against semi-honest server attack

In order to solve the so-called left-over problems [2]

$$\phi(\boldsymbol{A}\boldsymbol{R}_b\boldsymbol{R}_c),\ \phi(\boldsymbol{B}\boldsymbol{R}_a\boldsymbol{R}_c),\ \phi(\boldsymbol{C}\boldsymbol{R}_a\boldsymbol{R}_b)$$

it needs to use two-party scalar product protocols, where Merlin is one of the parties. A big difference between two-party protocol and three-party protocol is whether the semi-honest server involves in the procedures after the setup phase is completed.

In the three-party protocol, Alice has to perform the two-party protocol with Merlin in order to compute $k_a = \phi(\boldsymbol{A}\boldsymbol{M}_a)$ where $\boldsymbol{M}_a = \boldsymbol{R}_b\boldsymbol{R}_c$, and both $\boldsymbol{R}_b$ and $\boldsymbol{R}_c$ are known to Merlin. That means Alice needs to send $\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{R}_a$ to Merlin. Since Merlin knows $\boldsymbol{R}_a$, he can easily recover the private diagonal matrix $\boldsymbol{A}$ and the corresponding private vector A. Likewise, Merlin can retrieve the private vectors B and C.

To resist the semi-honest server attack, it suggests that the role of commodity server Merlin could be jointly played by many semi-honest parties. For instance, there are ℓ semi-honest servers, Server$_1$, Server$_2$, ..., Server$_\ell$, where $\ell > 3$. None of them can solely access $\boldsymbol{R}_a, \boldsymbol{R}_b, \boldsymbol{R}_c$. In this case, they need to collaboratively and securely compute the diagonal matrix

$$\boldsymbol{M}_a = \boldsymbol{R}_b\boldsymbol{R}_c \tag{3}$$

But we find the above problem is just a new secure ℓ-party computation problem, which is more intractable than the original 3-party computation problem.

It also suggests that (page 1064, [2]): "*a sufficient level of trust can be achieved to minimize the risk of this attack by enforcing the commodity server to act as an honest party, not just semi-honest.*" But we find the argument is self-contradictory. If the server is full-honest, not semi-honest, the original protocol becomes unnecessary. Actually, in this case any party-i can send $\hat{\boldsymbol{D}}_i = \boldsymbol{D}_i + \boldsymbol{R}_i$ to the honest server, who then retrieves $\boldsymbol{D}_i$ using $\boldsymbol{R}_i$. After all vectors are collected, the server computes the final scalar product and securely sends the result to any target user.

3.3 Exponential complexity

In the proposed n-party protocol, the party-I needs to compute

$$\begin{aligned}
&\phi(\boldsymbol{D}_1\boldsymbol{R}_2\boldsymbol{R}_3),\ \phi(\boldsymbol{D}_1\boldsymbol{R}_2\boldsymbol{R}_4),\ \cdots,\ \phi(\boldsymbol{D}_1\boldsymbol{R}_2\boldsymbol{R}_n);\\
&\phi(\boldsymbol{D}_1\boldsymbol{R}_2\boldsymbol{R}_3\boldsymbol{R}_4),\ \phi(\boldsymbol{D}_1\boldsymbol{R}_2\boldsymbol{R}_3\boldsymbol{R}_5),\ \cdots;\\
&\qquad\qquad \cdots\\
&\phi(\boldsymbol{D}_1\boldsymbol{R}_2\boldsymbol{R}_3\boldsymbol{R}_4\cdots\boldsymbol{R}_n)
\end{aligned}$$

That means any party needs to perform $2^{n-1} - n$ sub-protocols. The complexity increases exponentially with the number n. Therefore, the n-party protocol cannot be put into practice.

4 Conclusion

We show that the Daalen et al.'s scalar product protocol is insecure against semi-honest server attack. Its massive complexity is a big issue for practical implementation. The findings in this note could be helpful for the future work on designing such protocols.

References

[1] W. Du and Z. Zhan: Building decision tree classifier on private data, in *Proc. IEEE Int. Conf. Privacy Secur. Data Mining*, 2002, pp. 1-8.

[2] F. Daalen, L. Ippel, A. Dekker, and I. Bermejo: Privacy Preserving n-Party Scalar Product Protocol, *IEEE Trans. Parallel Distributed Syst.*, 34(4), pp. 1060-1066, 2023.