Conference Name: International Conference on Science & Technology, 09-10 September 2025, Rome
Conference Dates: 09-Sep- 2025 to 10-Sep- 2025
Conference Venue: Courtyard by Marriott Rome Central Park, Via Moscati 7 •00168 Rome • Italy
Appears in: MATTER: International Journal of Science and Technology (ISSN 2454-5880)
Publication year: 2025

Hwan Kim et.al, 2025

Volume 2025, pp. 56-66

DOI- https://doi.org/10.20319/stra.2025.5666

This paper can be cited as: Hwan Kim, K., Hyun Kim, S. & Yeong Lee, I.(2025). A Study on Secure Private Set Union Scheme in IOT Environment. International Conference on Science & Technology, 09-10 September 2025, Rome. Proceedings of Scientific and Technical Research Association (STRA), 2025, 56-66

A STUDY ON SECURE PRIVATE SET UNION SCHEME IN IOT ENVIRONMENT

Ki-Hwan Kim
Soonchunhyang University, Asan, South korea,
20247089@sch.ac.kr

Su-Hyun Kim
Soonchunhyang University, Asan, South korea,
kimsh@sch.ac.kr

Im-Yeong Lee
Soonchunhyang University, Asan, South korea,
imylee@sch.ac.kr

Abstract

In the IOT environment, network edges have data, but a centralized server may need to obtain the union from edges for efficient data access. PSU is a cryptographic primitive that allows protocol parties to compute the union of their private datasets without revealing any extra information. Traditional PSU protocols presuppose that all parties must input their private datasets. This assumption doesn't hold in some scenarios where an inputless Third party needs to get the union. For instance, a regulatory organization may need to get the union of patient data from hospitals for statistical analysis, without inputting any dataset. We propose a novel TP-PSU, specifically designed for a setting with three parties and an inputless Third party. Our protocol enables the

Third party to compute the union, while preventing the leakage of any other extra information. This includes protecting the origin and duplication of each data item across edge nodes, thus maintaining privacy in the IoT environment.

Keywords:
Data Privacy, Third Party, Private Set Union, Random Oblivious Transfer, Multi-Key Encryption

1. Introduction

In an IOT environment, each network edge holds its own data, and a centralized server may need to aggregate this data for analytics without leaking any other information (Nguyen, D. T. et al., 2021). Private Set Union (PSU) is a privacy-preserving cryptographic primitive that enables two or more parties to compute the union of their private sets without revealing any extra information (Jia, Y. et al, 2022). This primitive finds numerous practical applications, such as performing a full-join on database for private sets or managing IP blacklists from organizations for cyber risk assessment (Kolesnikov, V. et al., 2019). The PSU has also been extended to the Multiparty Private Set Union (MPSU), where multiple parties input their sets to compute the union (Liu, X. et al., 2023; Gao, J. et al., 2023).

A fundamental assumption in traditional PSU protocols is that all parties must input their private sets in protocol. However, this model cannot be used for some scenarios where an inputless Third party is required to get the union of the other parties' sets. This situation could arise in the IoT environment where centralized server collects and intergrates data from other network edges. This paper introduces a new Third party Private Set Union (TP-PSU), which allows an inputless Third party to securely compute the union of datasets provided by other parties while preventing any extra information leakage. The main contributions of our work are as follows:

- **Computation of the union of three sets by a Third party** : We propose a TP-PSU protocol specifically designed for a setting with three parties, each providing a private dataset. Our protocol ensures that a designated Third party (Third party corresponds to a centralized server, while the other parties represent network edges), who provides no input, can correctly obtain the union of these three sets.
- **Prevention of extra information leakage** : Our protocol guarantees that the Third party learns nothing beyond the final union set. Specifically, it is prevented from learning sensitive information such as (i) the origin of each element (i.e., which party inputted it) and (ii) which elements, if any, were duplicated across the parties' input sets. Furthermore, the parties themselves do not learn any extra information about each other's set.

2. Related Works

In this section, we review prior work on both Traditional PSU and MPSU.

2.1 Standard PSU Protocols

In the standard two-party PSU setting, a Sender and a Receiver compute the union of their private sets, with the result being delivered to the Receiver. The security requirements dictate that the Receiver must not learn any information beyond the union itself, such as the intersection or its cardinality, while the Sender must learn nothing at all. The work of (Kolesnikov, V. et al., 2019) introduced a scalable PSU protocol based on Oblivious Transfer (OT) and Reverse Private Membership Test (RPMT). However, their protocol potentially leaks the intersection to both the Sender and the Receiver during the RPMT phase. Although the subsequent work by Zhang et al. (Zhang, C. et al., 2022) proposed a Multi-Query RPMT (MQ-RPMT) to prevent this leakage to the Sender, the Receiver can still infer the intersection from the RPMT output bits (The RPMT bits mean the Private set intersection cardinality). The protocol by (Jia, Y. et al., 2024) enhances security by preventing the leakage of the intersection from the information during the execution.

2.2 MPSU Protocols

In the MPSU (the number of parties : n), typically $n-1$ Senders and one Receiver compute the union of their datasets, with the Receiver obtaining the result. A key security requirement is that the Receiver must not learn the provenance of each element in the union (i.e., which sender contributed it). (Liu, X. et al., 2023) proposed a protocol that satisfies these requirements, but it does not provide security if the Receiver colludes with other participants. To address this, (Gao, J. et al., 2023) introduced a public-key based MPSU protocol using Membership Oblivious Transfer (MOT) and a multi-key cryptosystem, which remains secure even in the presence of such collusion. More recently, (Dong, M. et al., 2024) proposed an MPSU protocol that satisfies all the security requirements while achieving optimal linear complexity in both communication and computation.

3. Preliminaries

In this section, we describe the ideal functionality of our TP-PSU and the functionality cryptographic primitives that serve as the building blocks for our TP-PSU protocol. In this paper we omit the detailed protocol process of the primitives.

3.1 TP-PSU Functionality

The ideal functionality of our TP-PSU $\mathcal{F}_{tp-psu}$ is shown in Figure 1. In a setting with three parties, P_1, P_2, and P_3, who input their private sets X_1, X_2, and X_3 respectively, the protocol allows a designated inputless Third party, TP, to get the union $X_1 \cup X_2 \cup X_3$. The protocol must ensure that TP learns nothing beyond the union itself, and no other extra information is leaked to any party.

3.2 SSPMT functionality

Secret-Sharing Private Membership Test (SSPMT) funtionality $\mathcal{F}_{sspmt}$ is shown in Figure 2, is a cryptographic primitive where a Sender $\mathcal{S}$ and a Receiver $\mathcal{R}$ can test for membership of the Sender's element in the Receiver's set. In here, $\mathbb{F}_{2^q}$ is a finited field. The result is output as a secret-shared value between the two parties. Specifically, if the element is a member, the parties receive secret shares, e_0 and e_1, of the value 1. Otherwise, they receive shares of 0. This functionality can be realized using the techniques presented by (Pinkas, B. et al., 2019).

Parameters : Party P_1, P_2, P_3, set size n and Third party TP and the bit length of element q

Funtionality :

- Wait for an input $X_{i\in[3]} = \{x_{i,1}, \dots, x_{i,n}\} \subset \mathbb{F}_{2^q}$ from P_1, P_2 *and* P_3, respectively
- Give output $X_1 \cup X_1 \cup X_3$ to Third party TP

Figure 1: *Third party Private Set Union functionality* $\mathcal{F}_{tp-psu}$

Parameters : Sender $\mathcal{S}$, Receiver $\mathcal{R}$ and set size n and the bit length of element q

Funtionality :

- Wait for an input element $x \in F_{2^q}$ from $\mathcal{S}$
- Wait for an input set $Y = \{y_1, \dots, y_n\} \subset \mathbb{F}_{2^q}$ from $\mathcal{R}$
- Give $e_0 \in \{0,1\}$ to $\mathcal{S}$ and $e_1 \in \{0,1\}$ to $\mathcal{R}$, where $e_0 \oplus e_1 = 1$ if and 0 otherwise.

Figure 2: *Secret Sharing Private Memembership Test functionality* $\mathcal{F}_{sspmt}$

Parameters : Sender $\mathcal{S}$, Receiver $\mathcal{R}$

Funtionality :

- Wait for an choice bit $b \in \{0,1\}$ from $\mathcal{R}$
- Sample to strings $(m_0, m_1) \leftarrow F_{2^q}$
- Give (m_0, m_1) to $\mathcal{S}$ and give m_1 to $\mathcal{R}$

Figure 3: *Random oblivious Transfer functionality* $\boldsymbol{\mathcal{F}_{rot}}$

3.3 ROT Functionality

Random Oblivious Transfer (ROT) functionality $\mathcal{F}_{rot}$ shown in Figure 3, is a variant of Oblivious Transfer. In this protocol, a Sender $\mathcal{S}$ holds two random messages (m_0, m_1), and a Receiver $\mathcal{R}$, with a choice bit b, obtains one of the two messages m_b. The security guarantees that the Receiver learns nothing about the message it didn't select, and the Sender learns nothing about the Receiver's bit b. This functionality is based on (Rabin, M. O., 2005).

3.4 Multi-Key Encryption

Multi-key encryption is used in previous MPSU studies (Gao, J. et al., 2023) and (Dong, M. et al., 2024). It is an ElGamal based cryptosystem but we omit the detailed process on this paper. (The detailed process is written in above MPSU papers.) We defined as five PPT algorithm as follows. As an example, we suppose that three protocol parties P_1, P_2 and P_3.

- $\boldsymbol{KeyGen}(\mathbf{1}^{\boldsymbol{\kappa}})$: As key generation algorithm, for $i \in [3]$, each P_i generates secret key sk_i and public key pk_i. And they obtain common public key $pk = Combine(pk_1, pk_2, pk_3)$. In here κ is computational security parameter.
- $\boldsymbol{Enc_{pk}(m)}$: As encryption algorithm, one of P_i can encrypt a message m with pk. Finally it can obtains a ciphertext ct.
- $\boldsymbol{ReEnc_{pk}(ct)}$: As re-encryption algorithm, one of P_i can re-encrypt a ciphertext ct with pk. Finally it can obtains a ciphertext ct'.
- $\boldsymbol{ParDec_{sk_i}(ct')}$: As partial decryption algorithm, one of P_i can partial decrypt a ciphertext ct' with his own sk_i. Finally it can obtains partical decrypted ciphertext ct''.

- $\boldsymbol{Dec_{sk_i}(ct'')}$: As decryption algorithm, one of P_i can decrypt a ciphertext ct''. Finally it can obtains a message m. This requires all P_j $(j \neq i)$ partially decrypt the ciphertext in advance.

4. Our TP-PSU Protocol

In this section, we propose TP-PSU protocol, which enables an inputless Third party to compute the union of three private sets. As notation, we denote each party P_1, P_2 and P_3 inputs their set $X_{i\in[3]} \subset \mathbb{F}_{2^q}$ (set size: n), respectively. The Third party is TP and it has a set $U = \emptyset$. The functionalities of SSPMT and ROT are $\mathcal{F}_{sspmt}$ and $\mathcal{F}_{rot}$, respectively. $\perp$ is a special dummy value. An overview of the protocol is shown in Figure 4, and the detailed steps are as follows.

- **Key generation step** : All parties call the key generation algorithm $KeyGen(1^\kappa)$, excluding TP. for $i \in [3]$, each P_i generates a pair of secret key and public key (sk_i, pk_i) and common public key pk.
- **SSPMT invocation step** : First, P_1 invokes n times $\mathcal{F}_{sspmt}$ with $P_{j\in[2,3]}$. P_1 acts as Receiver and P_j acts as Sender. Then, P_1 obtains the bits $(e_{1,2}^1, \dots, e_{1,2}^n)$ and P_j obtains $(e_{j,1}^1, \dots, e_{j,1}^n)$. Next, P_2 invokes n times $\mathcal{F}_{sspmt}$ with P_3. P_2 acts as Receiver and P_3 acts as Sender. Then, P_2 obtains the bits $(e_{2,3}^1, \dots, e_{2,3}^n)$ and P_3 obtains $(e_{3,2}^1, \dots, e_{3,2}^n)$.
- **Set encryption step** : For $i \in [3]$, each P_i encrypts their private sets X_i with the encryption algorithm as $Enc_{pk}(X_i) = X_i'$. Then, P_1 sends all elements of X_1' to TP and TP inserts all of the ciphertexts into U.
- **ROT invocation step (1)** : First, P_1 invokes n times $\mathcal{F}_{rot}$ with $P_{j\in[2,3]}$. P_1 acts as Receiver and P_j acts as Sender. For $t \in [n]$, P_1 obtains strings $r_{1,j}^t = r_{j,1,e_{1,j}^t}^t$ and P_j obtains $\left(r_{j,1,0}^t, r_{j,1,1}^t\right)$ as $\mathcal{F}_{rot}$ output. And P_j sends $(u_{j,1,e_{j,1}^t}^t, u_{j,1,e_{j,1}^t\oplus1}^t) = \left(Enc_{pk}(x_{j,t}) \oplus r_{j,1,e_{j,1}^t}^t, Enc_{pk}(\perp) \oplus r_{j,1,e_{j,1}^t\oplus1}^t\right)$ to P_1. Then, P_1 chooses $ct_{1,j}^t = u_{2,1,e_{1,j}^t}^t \oplus r_{1,j}^t$, sends $ct_{1,2}^t$ to TP and $ct_{1,3}^{t\prime} = ReEnc_{pk}(ct_{1,3}^t)$ to P_3. TP inserts all of the ciphertexts from P_1 into U.
- **ROT invocation step (2)** : Next, P_2 invokes n times $\mathcal{F}_{rot}$ with P_3. P_2 acts as Receiver and P_3 acts as Sender. For $t \in [n]$, P_2 obtains strings $r_{2,3}^t = r_{3,2,e_{2,3}^t}^t$ and P_3 obtains

$\left(r^t_{3,2,0}, r^t_{3,2,1}\right)$ as $\mathcal{F}_{rot}$ output. And, P_3 sends $(u^t_{3,2,e^t_{3,2}}, u^t_{3,2,e^t_{3,2}\oplus 1}) = \Big(ct^{t\prime}_{1,3} \oplus r^t_{3,2,e^t_{j,1}}, Enc_{pk}(\perp) \oplus r^t_{3,2,e^t_{j,1}\oplus 1}\Big)$ to Then, P_2 obtains $ct^t_{2,3} = u^t_{3,2,e^t_{2,3}} \oplus r^t_{2,3}$, sends them to TP. And TP inserts all of the ciphertexts from P_2 into U.

- **Set decryption step** : TP generates a key sk_{tp} and re-encrpyts set U. (sk_{tp} is a secret key used for both encryption and decryption.) Next, TP shuffles the set U sends it to P_1. Next, P_1 partially decrypts the set as $U' = ParDec_{sk_1}(U)$, shuffles and sends it to P_2. Next, P_2 partially decrypts the set as $U'' = ParDec_{sk_2}(U')$, shuffles and sends it to P_3. Next, P_3 partially decrypts the set as $U''' = ParDec_{sk_3}(U'')$, shuffles and sends it to TP. Next, TP decrypts the set as $R = ParDec_{sk_e}(U''')$ and removes all dummy $\perp$ in R. Finally TP can obtain the union $X_1 \cup X_2 \cup X_3$.

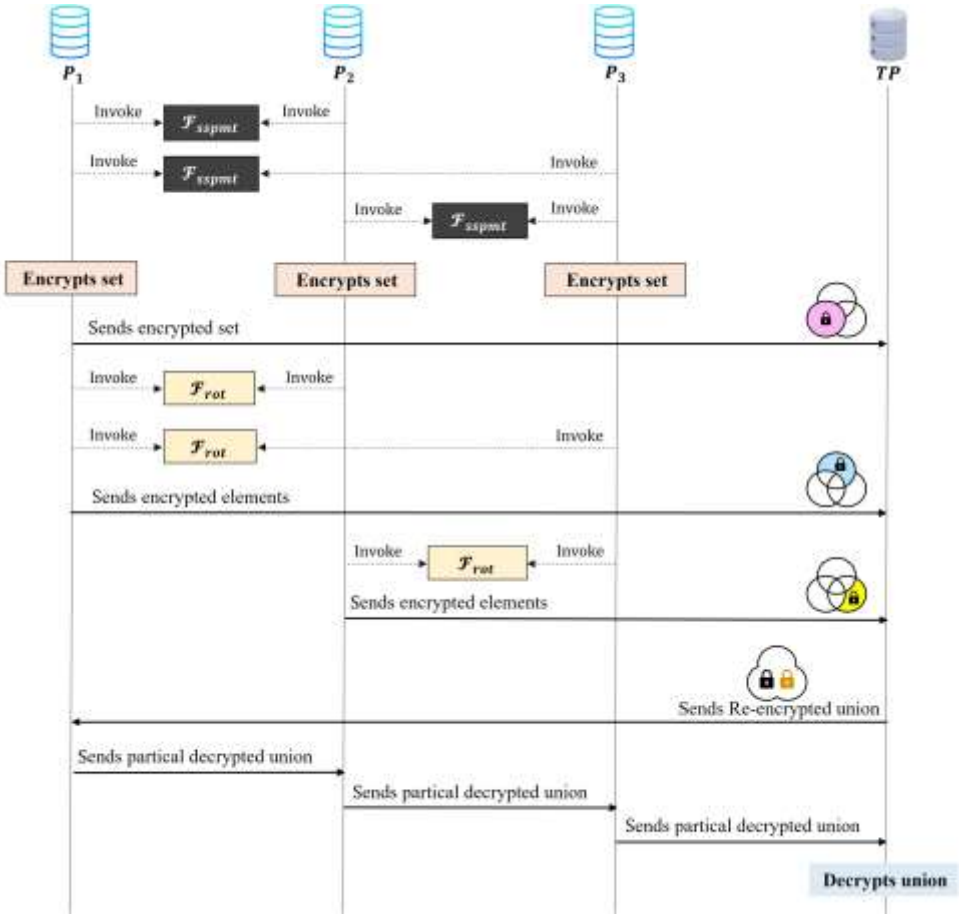

Figure 4: *An overview of our TP-PSU protocol*

Table 1. *Analysis of proposed TP-PSU with other state-of-the-art PSU*

	(Jia, Y. et al., 2024)	**(Dong, M. et al., 2024)**	**Our TP-PSU**
Number of parties	Two parties	Not limited	Three parties
Leakage of intersection	Secure	Secure	Secure
Leakage of elements origin	Not considered	Secure	Secure
Applicability in inputless Third party scenarios	Not provided	Not provided	Provided

5. Analysis of the proposed TP-PSU

In this section, we provide a comparative analysis of our proposed TP-PSU protocol with existing state-of-the-art PSU studies. A summary of the overall analysis is shown in Table 1 and detailed analysis results are as follows:

- **Number of Parties** : Standard PSU (Jia, Y. et al., 2024) is confined to a two-party setting and MPSU (Dong, M. et al., 2024) support an arbitrary number of parties. However, they do not accommodate scenarios involving an inputless Third Party. Our TP-PSU protocol is specifically designed for three parties (excluding Third party) and enables a Third party to compute the union. In terms of scalability, this is admittedly a step back compared to general MPSU, but it addresses a distinct and practical use case.
- **Leakage of intersection** : During the $\mathcal{F}_{rot}$ phase, ciphertexts of duplicated elements are replaced with encrypted dummy values $\perp$. As a result, when TP decrypts the set, duplicate elements are replaced with indistinguishable dummy values, ensuring that TP cannot identify intersection elements within the result.
- **Leakage of elements origin** : To prevent the Third party from tracing elements back to their origin, each party P_1, P_2 and P_3 shuffles the encrypted union in partial decryption step. These operations make it infeasible for TP to find the elements origin because it has no idea about other parties' shuffling function.
- **Applicability for inputless Third party scenario** : Prior PSU and MPSU protocols require one of the data-owning parties to receive the result, leaving TP reliant on that party to obtain the union. This introduces a risk of information misuse or manipulation. In contrast, our protocol is explicitly designed so that TP directly computes and decrypts the

final union without needing to trust any input-holding party. This design is well-suited for practical cases in which an inputless Third party must securely compute the union of sets.

6. Conclusions

PSU is a technique for securely computing union of private sets. However, a significant limitation of existing PSU studies is their inapplicability to scenarios where an inputless Third party must obtain the union. To address this limitation, we proposed TP-PSU, a novel protocol that provides privacy guarantees for a Third party computing the union of three private sets. This is especially useful in the IoT environment, where a centralized server may collect data from network edges. Our protocol enables an inputless Third party to securely compute the union without revealing any extra information. This makes our solution well-suited for practical IoT scenarios, such as when a centralized server needs to obtain union data from multiple network edges. Although our TP-PSU is focused on three parties setting (excluding Third party), we view this as a foundational step. Future work could extend this scheme to support a general n-party scenario, where a Third party securely computes the union of multiple sets.

Acknowledgements

This research was supported by BK21 FOUR (Fostering Outstanding Universities for Research) (No.: 5199990914048), and this work is the result of a commissioned research project supported by the affiliated institute of ETRI (2025-074).

References

Dong, M., Chen, Y., Zhang, C., & Bai, Y. (2024). Breaking free: Efficient multi-party private set union without non-collusion assumptions. *arXiv preprint arXiv:2406.07011*.

Gao, J., Nguyen, S., & Trieu, N. (2023). Toward A practical multi-party private set union. *Cryptology ePrint Archive*.

Jia, Y., Sun, S. F., Zhou, H. S., & Gu, D. (2024). Scalable private set union, with stronger security. In *33rd USENIX Security Symposium (USENIX Security 24)* (pp. 6471-6488).

Jia, Y., Sun, S. F., Zhou, H. S., Du, J., & Gu, D. (2022). Shuffle-based private set union: Faster and more secure. In *31st USENIX Security Symposium (USENIX Security 22)* (pp. 2947-2964).

Kolesnikov, V., Rosulek, M., Trieu, N., & Wang, X. (2019, November). Scalable private set union from symmetric-key techniques. In *International conference on the theory and application of cryptology and information security* (pp. 636-666). Cham: Springer International Publishing.

Liu, X., & Gao, Y. (2023, December). Scalable multi-party private set union from multi-query secret-shared private membership test. In *International conference on the theory and application of cryptology and information security* (pp. 237-271). Singapore: Springer Nature Singapore.

Nguyen, D. T., & Trieu, N. (2022). MPCCache: Privacy-preserving multi-party cooperative cache sharing at the edge. In *International Conference on Financial Cryptography and Data Security* (pp. 80–99). Cham: Springer International Publishing.

Pinkas, B., Schneider, T., Tkachenko, O., & Yanai, A. (2019). Efficient circuit-based PSI with linear communication. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III 38* (pp. 122-153). Springer International Publishing.

Rabin, M. O. (2005). How to exchange secrets with oblivious transfer. *Cryptology ePrint Archive*.

Zhang, C., Chen, Y., Liu, W., Zhang, M., & Lin, D. (2022). Optimal Private Set Union from Multi-Query Reverse Private Membership Test. *IACR Cryptol. ePrint Arch.*, *2022*, 358.