Sectric: Towards Accurate, Privacy-preserving and Efficient Triangle Counting

Minze Xu
State Key Laboratory for Novel Software Technology
Nanjing University
cnmzxu@gmail.com

Zhentai Xie
State Key Laboratory for Novel Software Technology
Nanjing University
zhentaixie@smail.nju.edu.cn

Zhibin Wang
State Key Laboratory for Novel Software Technology
Nanjing University
wzbwangzhibin@gmail.com

Guangzhan Wang
State Key Laboratory for Novel Software Technology
Nanjing University
502023330057@smail.nju.edu.cn

Longbin Lai
Alibaba Group
longbin.lailb@alibaba-inc.com

Yuan Zhang
State Key Laboratory for Novel Software Technology
Nanjing University
zhangyuan@nju.edu.cn

Chen Tian
State Key Laboratory for Novel Software Technology
Nanjing University
tianchen@nju.edu.cn

Sheng Zhong
State Key Laboratory for Novel Software Technology
Nanjing University
zhongsheng@nju.edu.cn

ABSTRACT

Graph data analysis, particularly local triangle counting, plays a pivotal role in deciphering complex relationships within graph data. This method is invaluable across diverse fields such as social networks, transportation, and cybersecurity. However, this process often involves handling sensitive information, necessitating that the relationship between any two nodes is considered private. Differential privacy (DP) is a formal model to address privacy concerns and can be categorized into two types: the central DP (CDP) model, which achieves better result accuracy, and the local DP (LDP) model, which does not assume a trusted server. To bridge the gap between the two models, we propose Sectric, a server-aided crypto-assisted local triangle counting protocol, in this paper. It can achieve the same result accuracy with the same privacy budget as the CDP model without assuming a trusted server. Sectric also explores a new approach in crypto-assisted graph data analysis algorithms that represents a node's neighbors using a set instead of an adjacency vector, and successfully achieves higher efficiency compared to other crypto-assisted solutions. We also conduct theoretical and empirical evaluations to demonstrate that Sectric achieves the design principles.

PVLDB Reference Format:
Minze Xu, Zhentai Xie, Zhibin Wang, Guangzhan Wang, Longbin Lai, Yuan Zhang, Chen Tian, and Sheng Zhong. Sectric: Towards Accurate, Privacy-preserving and Efficient Triangle Counting. PVLDB, 18(10): 3382 - 3395, 2025.
doi:10.14778/3748191.3748202

Proceedings of the VLDB Endowment, Vol. 18, No. 10 ISSN 2150-8097.
doi:10.14778/3748191.3748202

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/zhentaixie/Sectric.

1 INTRODUCTION

Graph data analysis is pivotal in unraveling complex relationships and patterns in graph data, making it a useful tool in various fields such as social networks [3, 10, 24, 38], transportation and logistics [30, 41, 50, 59], and cybersecurity [25, 61, 62]. In many applications, graph data are stored in a decentralized manner [5, 31], where each node knows its neighbors, while no central node has the full graph topology. In this setting, *local triangle counting*, which calculates the triangle counts containing a given node, is a fundamental graph analysis task. It is widely applied to tasks such as community detection, node importance evaluation, and network structure analysis [1, 6, 16, 32, 38, 45, 56, 57], all of which are relevant for applications like recommendation systems and fraud detection. We list some downstream tasks of local triangle counting in Table 1.

Privacy is another concern in the decentralized setting. In the broader trend of federated graph analytics, where users may wish to participate in collaborative computation without revealing their full neighborhood. This reflects increasing demand for privacy-aware computation models, particularly in regulated industries (e.g., financial networks, healthcare). Differential privacy (DP) is a formal model addressing this privacy concern. Prior differentially private solutions can be mainly categorized into two types: adapting the central DP (CDP) model or the local DP model (LDP). The CDP model assumes a trusted server to calculate the triangle counts and adds a small noise to the final result[1]. The LDP model eliminates the trust assumption on the server, but has to add more noise in the

[1]In the CDP model, the server knows the complete and accurate topology of the graph, including all nodes and the connection edges to their neighbors.

Table 1: Downstream tasks of local triangle counting.

Downstream tasks	Related works	Applications
Local Clustering Coefficient	[6, 19, 58]	Spam detection
Weighted Community Clustering	[52, 63]	Community Detection
K-Truss Decomposition	[26, 53]	Community Detection
Triangle Centrality Measures	[7, 13]	Node Importance Evaluation

calculating process with the same privacy budget. Thus, the CDP model has better result accuracy, and the LDP model aligns better with the decentralized setting.

To bridge the gap between the two models, we notice the emerging crypto-assisted approach in other graph analysis tasks [35, 54] and explore its application in privacy-preserving local triangle counting. This approach makes graph nodes interact with graph nodes using cryptographic primitives to finally obtain the analysis result. Compared to the LDP model, the crypto-assisted approach provides crypto-level privacy protection and has an accurate analysis result due to the use of cryptographic tools.

Following this framework, we design Sectric, a server-aided crypto-assisted triangle counting protocol. The first challenge we meet is the high overheads incurred by directly applying existing techniques. For example, if we directly adapt techniques from CARGO [35] or MAGO [54] to privacy-preserving local triangle counting, it will bring $O(|\mathcal{V}|^2)$ overheads($|\mathcal{V}|$ denotes the number of graph nodes), which are unsatisfactory on large graphs. We observe that the root of the high overheads lies in operating on the graph's adjacency matrix. To reduce the overheads, we explore the usage of adjacency sets, instead of adjacency vectors, to represent node adjacencies. To implement this idea, we design a novel cryptographic tool named *Three-Party Private Set Membership Test (3PPSMT)*, rather than using secret sharing as CARGO and MAGO. This primitive reduces the overheads of counting two graph nodes' common neighbors from $O(|\mathcal{V}|)$ to $O(d_{max})$ compared to secret sharing(d_{max} denotes the maximum degree of graph nodes). With this primitive, Sectric introduces only $O(d_{max}|\mathcal{V}|)$ overheads in privacy-preserving local triangle counting. It makes Sectric more suitable for analyzing sparse graphs (i.e., d_{max} is sublinear to $|\mathcal{V}|$).

Meanwhile, Sectric enables the querier to calculate the accurate counting result. It guarantees utility but may also reveal graph nodes' adjacency relationship. To prevent this privacy leakage in the computation result, we also demonstrate that Sectric is compatible with the DP mechanism. Sectric allows adding noise subject to a given distribution, ensuring that the querier receives only the noisy result, while the server gains no information. The noise intensity is the same as that in the CDP model given the same privacy budget. In a nutshell, Sectric has the same result utility when providing the same privacy guarantee as the CDP model, and meanwhile requires no trusted server as the LDP model.

Furthermore, we also fully utilize the local view of graph nodes to reduce the number of required servers. Prior crypto-assisted graph data analysis solutions require two or more non-collusive servers to assist, while Sectric only requires one server. This requirement is easier to implement in practical applications.

The main contributions of this work can be summarized as:

- We design a novel local triangle counting protocol Sectric bridging the trust assumption and utility gap between the CDP and LDP models in the problem. Our solution shows that the same privacy and utility guarantee as the CDP can be achieved without requiring a trusted server.
- We explore a new approach in crypto-assisted graph data analysis algorithms that represents a node's neighbors using a set instead of an adjacency vector and reduces the overheads from $O(|\mathcal{V}|^2)$ to $O(d_{max}|\mathcal{V}|)$. We believe that this approach can also be adopted in other tasks to reduce the overheads and will further explore it in the future.
- We perform a comprehensive theoretical and empirical analysis of Sectric to demonstrate its privacy guarantees and performance. We also adapt the open-source implementation of a state-of-the-art work [35, 51] to local triangle counting as the baseline.

Paper Organization. The rest of this paper is organized as follows: In Section 2, we discuss the related works. Then, we define the problem in Section 3. In Section 4, we define the primitive $\mathcal{F}_{\mathsf{3PPSMT}}$ and propose the Π_{3PPSMT} protocol implementing this primitive. Based on this primitive, the construction of the Sectric protocol is presented in Section 5. Section 6 presents the experimental results, and Section 7 concludes this paper.

2 RELATED WORK

2.1 Privacy-Preserving Triangle Counting

The problem of privacy-preserving triangle counting has been an active area of research. Existing works can be broadly categorized into two groups based on whether they assume the existence of a trusted server or not: centralized model-based approaches and decentralized model-based approaches.

The centralized model assumes the existence of a trusted server that holds the entire graph. Ding et al. [11] achieve a balance between the accuracy of triangle counting and data privacy by selecting appropriate edge deletion strategies. Raskhodnikova et al. [28, 39] use randomized strategies to ensure that the published triangle counts do not accurately allow inferring the existence of any particular edge, while Kasiviswanathan et al. [29] have achieved this by projecting the graph with a limited degree threshold. However, these approaches require a fully trusted and central server, which can introduce privacy issues in many applications.

The solutions proposed by these works can provide privacy in the existence of a fully trusted central server. However, in scenarios where a trusted server is intractable to implement, their solutions cannot be directly applied.

Thus, many works have proposed a decentralized model, where the node set is still considered public knowledge, but the relationship between two nodes is only known to them and treated as their privacy. Sun et al. [49] propose a local differential privacy approach, where graph nodes locally perturb their adjacency vectors to protect the privacy of edges. However, their assumption that graph nodes have an extended local view, allowing them to see their 2-hop neighbors, introduces the data correlation problem [37], and is not applicable in most real-world cases. In the more realistic scenario where nodes can only see their immediate neighbors, Imola

Table 2: Comparison of Sectric with related works on local triangle counting.

Protocol	Privacy Model	Number of Servers	Expected l_2 Loss	Computation Overheads	Communication Overheads
ARRFull,ARROneNs,ARRTwoNs [22]	DP	1	$O(\frac{d_{max}^2}{(1-e^{-\epsilon})^2})$	$O(\|\mathcal{V}\|^2)$	$O(\|\mathcal{V}\|^2)$
WShuffle [23]	DP	2	$O(d_{max}^3)$	$O(\|\mathcal{V}\|^2)$	$O(\|\mathcal{V}\|^2)$
CARGO [35]	DP+Crypto	2	$O(\frac{1}{\epsilon^2})$	$O(\|\mathcal{V}\|^2)$	$O(\|\mathcal{V}\|^2)$
MAGO [54]	DP + Crypto	3	$O(\frac{1}{\epsilon^2})$	$O(\|\mathcal{V}\|^2)$	$O(\|\mathcal{V}\|^2)$
Sectric (ours)	Crypto	1	0	$O(d_{max}\|\mathcal{V}\|)$	$O(d_{max}\|\mathcal{V}\|)$
DPSectric (ours)	DP + Crypto	1	$O(\frac{1}{\epsilon^2})$	$O(d_{max}\|\mathcal{V}\|)$	$O(d_{max}\|\mathcal{V}\|)$

et al. [21, 22] utilize multiple rounds of interactions to upload and download perturbed edge information. While this approach can preserve privacy, it also introduces a non-negligible additive error, as highlighted by [12].

In the decentralized model, crypto-assisted solutions to triangle counting are also emerging in recent years. CARGO [35] utilizes a hybrid approach that combines additive secret sharing and differential privacy, allowing two untrusted servers to only see encoded values beyond other information. This approach enables graph nodes to add smaller amounts of noise when implementing differential privacy, thereby achieving better utility compared to [22]. Building on a similar approach, Imola et al. [23] introduce a trusted intermediate server with shuffling functionality. Another work, MAGO [54], which is based on lightweight secret sharing techniques, utilizes three servers from different trust domains working in coordination to improve the accuracy of triangle counting, and also detect whether malicious adversaries attempt to tamper with the statistical result.

A summary of the comparison between Sectric and other related works on local triangle counting is presented in Table 2.

2.2 Crypto-Assisted Graph Analytics

Cryptographic techniques are widely applied in protecting database privacy [4, 15, 60, 67]. Crypto-assisted solutions are also emerging in other graph analytics tasks [8, 33, 35, 54, 64].

Some works enable graph nodes to securely contribute their local views on a decentralized social graph for spectral analytics. Sharma et al. [47] utilize homomorphic encryption to protect the privacy of graph edges, allowing distributed data owners to interact with cloud-based programs while keeping their data private from the cloud service provider. PrivGED [55] employs secret sharing to encrypt the elements in local view vectors, enabling privacy-preserving eigen-decomposition analytics over decentralized social graphs while safeguarding graph nodes' social relationships. However, these studies focus on different analytical tasks than our work on local triangle counting.

Another line of research leverages cryptographic techniques for privacy-preserving epidemiological analysis, such as analyzing transmission chains or clusters to predict infection rates using contact data stored on mobile devices. RIPPLE [18, 20] enables realistic simulations on the actual person-to-person social contact graph, utilizing a set of semi-honest non-colluding MPC servers to facilitate communication among participants. Colo [34] introduces a protocol that guards against malicious device behavior using random masks, efficient commitments, and range proofs, ensuring that devices only learn their own node, edge, and topology data, while the analyst only learns the query result. However, these methods are not directly applicable to our local triangle counting problem.

There is also a research direction focusing on collaborative graph analytics, where each client possesses a local subgraph with multiple nodes and edges. Araki et al. [2] propose a secure shuffling method for a 3-server setting with an honest majority, implementing algorithms like breadth-first search and maximal independent set. Guan et al. [17] design a scheme for two data owners to jointly respond to a subgraph matching query without disclosing their graph datasets to each other. FEAT [36] has a central server that collects subgraph data from clients using private set union, aggregates them into a noisy global graph, and performs triangle counting. Oryx [66] can detect cycles of various lengths on a multi-party federated graph while preserving topological secrecy. Pang et al. [40] design a scheme based on structured encryption and private set intersection cardinality techniques. They provide server tokens to queriers to query the butterfly counts of specific nodes or edges. However, the assumptions in these studies differ from our scenario, where graph nodes only have a local perspective.

3 PROBLEM DEFINITION

We first introduce some notations used in this paper. For a positive integer N, $[N]$ denotes the set $\{1, 2, \ldots, N\}$, and $\mathbb{Z}_N$ represents the group modulo N. Given a set $\mathcal{X}$, $x \xleftarrow{\$} \mathcal{X}$ indicates that x is uniformly selected from $\mathcal{X}$.

3.1 Local Triangle Counting

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph, where $\mathcal{V}$ and $\mathcal{E}$ represent the set of nodes and edges, respectively. Two nodes $u, v \in \mathcal{V}$ are adjacent if $(u, v) \in \mathcal{E}$. We consider undirected graphs, so for any two nodes $u, v \in \mathcal{V}$, $(u, v) \in \mathcal{E}$ if and only if $(v, u) \in \mathcal{E}$. Without loss of generality, we assume the nodes in $\mathcal{V}$ are indexed from 1 to $|\mathcal{V}|$.

The notion of local triangle sets is defined in Definition 1. Intuitively speaking, a triangle in the graph is a subgraph consisting of three vertices and three edges, forming a cycle of length three and the local triangle set Δ_u of a node u is the set of all triangles containing u. With the notion of local triangle set, the local triangle counting problem can be stated as calculating $|\Delta_u|$ given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a node $u \in \mathcal{V}$.

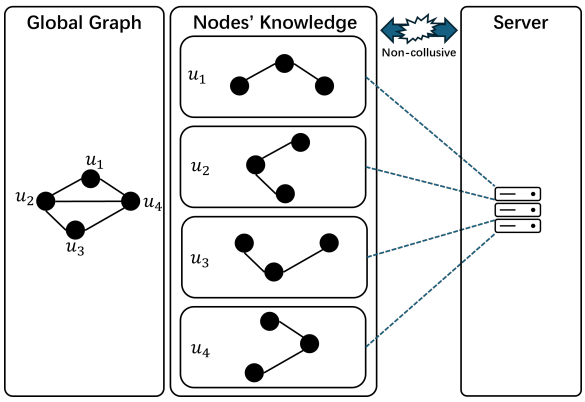


Figure 1: An illustrative example of the Sectric system model and threat model. The system consists of graph nodes and a server, with communication channels (dashed lines) between the nodes and the server. The server and nodes are semi-honest parties, and the server will not collude with nodes.

Definition 1 (Local triangle set)**.** Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the local triangle set of a node $u \in \mathcal{V}$ is defined as:

$$\Delta_u = \{\{u, v, w\} \subset \mathcal{V} : (u, v), (u, w), (v, w) \in \mathcal{E}\}.$$

We also define the notion of a graph node's neighbor set in Definition 2. The neighbor set N_u of a node u denotes the set of all nodes adjacent to u.

Definition 2 (Neighbor set)**.** Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we define the neighbor set of a node $u \in \mathcal{V}$ as

$$N_u = \{v \in \mathcal{V} : (u, v) \in \mathcal{E}\}.$$

The following theorem establishes the relationship between the local triangle sets and the neighbor sets, which serves as the foundation for our Sectric protocol.

Theorem 1. *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, for any node $u \in \mathcal{V}$, we have*

$$|\Delta_u| = \frac{1}{2} \sum_{v \in N_u} |N_u \cap N_v|.$$

Proof. We know that the number of triangles containing node u is given by: $|\Delta_u| = \sum_{v \in N_u} \sum_{w \in N_u, w > v} I(v, w)$, where $I(v, w)$ is an indicator function that equals 1 if there is an edge between v and w, and 0 otherwise.

Additionally, we have: $|N_u \cap N_v| = \sum_{w \in N_u} I(v, w)$. Thus, we can express the sum as: $\sum_{v \in N_u} |N_u \cap N_v| = \sum_{v \in N_u} \sum_{w \in N_u} I(v, w)$.

In the calculation process, the positions of v and w are equivalent and interchangeable. Therefore, we can conclude that: $|\Delta_u| = \sum_{v \in N_u} \sum_{w \in N_u, w > v} I(v, w) = \frac{1}{2} \sum_{v \in N_u} |N_u \cap N_v|$. □

3.2 System Model

We design Sectric in a server-aided paradigm. The system model is illustrated in Figure 1. The system participants include the server $\mathcal{S}$ and the graph nodes. The adjacency relations between the graph nodes are represented by the graph edges. The server $\mathcal{S}$ establishes a communication channel with each graph node.

In the decentralized setting, the number of graph nodes is public knowledge, and each graph node $u \in \mathcal{V}$ is only aware of its neighbor set N_u (ref. Definition 2). The server $\mathcal{S}$ has no knowledge about the adjacency relationship between the graph nodes.

In Sectric, a graph node Q acts as the querier and requests the number of its local triangles. During the protocol execution, the querier Q interacts with the server $\mathcal{S}$ and its neighbors. Finally, the protocol outputs the number of Q's local triangles to Q.

3.3 Threat Model and Privacy Constraints

The threat model we consider in this work is the semi-honest model, which is commonly adopted in the context. It assumes that the graph nodes and the server $\mathcal{S}$ will follow the Sectric protocol, but may attempt to conjecture a given node's neighbor set in protocol execution. The threat model allows collusion of graph nodes, but requires that the server does not collude with any graph node.

Against this threat model, Sectric targets at protecting the privacy of the adjacency relations of graph nodes in protocol execution. We refer to this privacy constraint as preserving the "neighbor privacy" of graph nodes. To capture this privacy objective, we define a cryptographic-style privacy constraint in Definition 3. In the definition, "Simulation" refers to that the messages' distributions and the algorithm's outputs are computationally indistinguishable.

Definition 3 (Privacy Constraint)**.** Sectric is a privacy-preserving protocol if there exist polynomial-time algorithms $\mathcal{A}_\mathcal{S}$, $\mathcal{A}_Q$, and $\mathcal{A}_{\mathcal{V}'}$ such that the messages received by $\mathcal{S}$, the querier Q, and graph nodes $\mathcal{V}' \subset \mathcal{V}$ in the execution of Sectric can be simulated by $\mathcal{A}_\mathcal{S}(1^\lambda, 1^{|\mathcal{V}|}, \Delta_Q)$, $\mathcal{A}_Q(1^\lambda, 1^{|\mathcal{V}|}, \Delta_Q)$, and $\mathcal{A}_{\mathcal{V}'}(1^\lambda, 1^{|\mathcal{V}|}, \Delta_Q)$, respectively, where λ is the security parameter.

However, the privacy constraint only restricts the privacy leakage in participants' interactions during protocol execution. Sectric is designed to provide accurate counting result, and directly releasing the result is also possible to reveal adjacency relations of graph nodes. To provide higher privacy guarantee, Sectric can also add randomness before releasing the final result to provide edge-level DP. Below, we define the edge-level DP.

Definition 4 (ϵ-Edge CDP [43])**.** Let $\epsilon \geq 0$. A randomized algorithm $\mathcal{M}$ with domain $\mathcal{G}$ provides ϵ-Edge CDP if for any two neighboring Graph G, G' that differ in one edge and any $S \subset Range(\mathcal{M})$,

$$Pr[\mathcal{M}(G) \in S] \leq e^\epsilon Pr[\mathcal{M}(G') \in S],$$

where ϵ is the privacy budget and $Range(\mathcal{M})$ denotes the set of possible outputs of $\mathcal{M}$.

4 THREE-PARTY PRIVATE SET MEMBERSHIP TEST

4.1 Basic Primitives

We first introduce the cryptographic tools used in this work.

Random oracles. A random oracle is a theoretical construct used in the context of cryptography and complexity theory. It is defined by specifying query and image domains, and it responds with an element from the image domain for every query in the query domain. A random oracle $\mathcal{H}$ responds with a uniformly random value for every newly appeared query, and a fixed value for repeated queries.

Random oracles are typically implemented using cryptographic hash functions, such as SHA-2 or SHA-3.

Multi-point OPRF (mpOPRF). A pseudorandom function (PRF) is a cryptographic tool that emulates a random function. Specifying a key k, the outputs of a pseudorandom function $f_k(\cdot)$ on different inputs appear random to parties without knowing k. In practice, it is usually implemented with symmetric encryption algorithms, such as AES.

A mpOPRF protocol involves the sender and the receiver, which specify a key k and a series of inputs $\langle x_i \rangle_{i\in[n]}$, respectively. It allows the receiver to learn the outcomes $\langle f_k(x_i)\rangle_{i\in[n]}$ without revealing additional knowledge to the sender or the receiver. This functionality is denoted as $\mathcal{F}_{\mathsf{mpOPRF}}$.

Private Equality Test. Private equality test is a two-party protocol allowing two parties, say P_0 and P_1, to test the equality of their inputs, and secret-share the result. The state-of-the-art implementation of this functionality is proposed by Chandran et al. [9]. This functionality is denoted as $\mathcal{F}_{EQ}$

Oblivious Key-Value Store (OKVS). The notion of oblivious key-value store (OKVS) is first introduced by Garimella et al. [14] in the context of private set intersection (PSI). OKVS allows one to encode a set of key-value pairs into an encoding, and ensures that the original key-value pairs generating the encoding cannot be recovered from the encoding given that the encoded values are uniformly random.

Definition 5 (Key-Value Store [14])**.** A key-value store (KVS) is defined by specifying the key space $\mathcal{K}$ and the value space $\mathcal{V}$, together with two algorithms:

(1) $S \leftarrow E(A; R)$: The encoding algorithm takes a list of n key-value pairs $A = \langle (k_i, v_i)\rangle_{i\in[n]} \subset \mathcal{K}\times\mathcal{V}$ with distinct keys and the randomness R as inputs. It outputs an encoding $S \in \mathcal{V}^m \cup \{\perp\}$.
(2) $v \leftarrow D(S, k)$: The decoding algorithm takes the encoding $S \in \mathcal{V}^m$ and a key $k \in \mathcal{K}$ as the inputs. It outputs a value $v \in \mathcal{V}$.

Definition 6 (Expansion ratio)**.** Given a KVS scheme Π, if the encoding S storing n key-value pairs satisfies $S \in \mathcal{V}^m \cup \{\perp\}$, then the expansion ratio of Π is $\frac{m}{n}$.

Definition 7 (Obliviousness)**.** A KVS $\Pi = (E, D)$ defined on the key space $\mathcal{K}$ and value space $\mathcal{V}$ satisfies the condition of *obliviousness* if, for any two lists of n distinct keys $\langle k_i\rangle_{i\in[n]}$ and $\langle k'_i\rangle_{i\in[n]}$, and n values $\langle v_i\rangle_{i\in[n]} \xleftarrow{\$} \mathcal{V}^n$ drawn uniformly at random from $\mathcal{V}^n$, and for any polynomial-time algorithm $\mathcal{A}$,

$$|Pr[\mathcal{A}(S) = 1] - Pr[\mathcal{A}(S') = 1]|$$

is negligible, where

$$S \leftarrow E(\langle (k_i, v_i)\rangle_{i\in[n]}) \text{ and } S' \leftarrow E(\langle (k'_i, v_i)\rangle_{i\in[n]}).$$

In other words, the distributions of S and S' are computationally indistinguishable.

4.2 Protocol Construction

In this part, we propose our construction to implement the 3PPSMT primitive. The 3PPSMT primitive is a three-party functionality $\mathcal{F}_{\mathsf{3PPSMT}}$ that allows the querier Q to test whether an element is in the set provided by a set provider U. The set X and the element u are selected from a universe $\mathcal{U}$. The result, indicating whether $u \in X$ or not, is secret-shared between the querier Q and server $\mathcal{S}$.

Algorithm 1 The Π_{3PPSMT} protocol.

Parameters: The security parameter λ.
Involved parties and inputs:

- The server $\mathcal{S}$: no input.
- The querier Q: an element $u \in \{0,1\}^\lambda$.
- The set provider U: a set $X \subset \{0,1\}^\lambda$.

Cryptographic primitives:

- A PRF family $\{f_s\}$ and a protocol securely implementing the functionality $\mathcal{F}_{\mathsf{mpOPRF}}$ for $\{f_s\}$.
- An OKVS scheme $\Pi = (E, D)$ with $\mathcal{K} = \mathcal{V} = \{0,1\}^\lambda$.
- A private equality test protocol securely implementing the functionality $\mathcal{F}_{\mathsf{EQ}}$.

Preprocessing:

1: U uniformly samples $k_U \xleftarrow{\$} \{0,1\}^\lambda$ and evaluates $y_i = f_{k_U}(x_i)$ for all $x_i \in X$.
2: U encodes $S_U \leftarrow \Pi.E(\langle (x_i, y_i)\rangle_{x_i\in X})$, and sends all $\langle y_i\rangle_{x_i\in X}$ to $\mathcal{S}$.

Online phase:

3: $\mathcal{S}$ samples $r \xleftarrow{\$} \{0,1\}^\lambda, k_{\mathcal{S}} \xleftarrow{\$} \{0,1\}^\lambda$, and encodes $S_{\mathcal{S}} = \Pi.E(\langle (y_i, f_{k_{\mathcal{S}}}(y_i) + r \mod 2^\lambda)\rangle_{x_i\in X})$
4: U sends S_U to Q, and $\mathcal{S}$ sends $S_{\mathcal{S}}$ to Q.
5: Q obtains $y = \Pi.D(S_U, u)$ and $\hat{y} = \Pi.D(S_{\mathcal{S}}, y)$.
6: $\mathcal{S}$ and Q invoke $\mathcal{F}_{\mathsf{mpOPRF}}$ as the sender and the receiver with inputs $k_{\mathcal{S}}$ and y, respectively.
7: Supposing Q obtains y' as the output of $\mathcal{F}_{\mathsf{mpOPRF}}$, it evaluates $r' = \hat{y} - y' \mod 2^\lambda$.
8: $\mathcal{S}$ and Q invoke $\mathcal{F}_{\mathsf{EQ}}$ with inputs r and r', respectively, and have the outputs $b_{\mathcal{S}}$ and b_Q from $\mathcal{F}_{\mathsf{EQ}}$.

The Π_{3PPSMT} protocol is presented in Algorithm 1. In this protocol, the querier Q inputs an element $u \in \{0,1\}^\lambda$ and the set provider U inputs a set X. This protocol aims to test whether $u \in X$ and secret-shares the output between the querier Q and a server $\mathcal{S}$.

The Π_{3PPSMT} protocol consists of a querier-independent preprocessing phase and a querier-involved online phase. In the preprocessing phase, the server $\mathcal{S}$ and the set provider U need not interact with the querier Q. The querier Q and the query u are only involved in the online phase.

In the preprocessing phase, U first samples a PRF key k_U and maps the set elements $x_i \in X$ to $y_i = f_{k_U}(x_i)$ in Step 1. Then, it encodes the key-value pairs $\langle (x_i, y_i)\rangle_{i\in[|X|]}$ into an OKVS S_U. Once receiving y_i from U, $\mathcal{S}$ samples a random number r from $\{0,1\}^\lambda$ and another PRF key $k_{\mathcal{S}}$. In the following, it encodes the key-value pairs $\langle (y_i, (f_{k_{\mathcal{S}}}(y_i) + r \bmod 2^\lambda))\rangle_{i\in[|X|]}$ into another OKVS $S_{\mathcal{S}}$ in Step 3.

In the online phase, $\mathcal{S}$ and U first send $S_{\mathcal{S}}$ and S_U to Q, respectively, in Step 4. After receiving S_U and $S_{\mathcal{S}}$, Q decodes them to obtain $y = \Pi.D(S_U, u)$ and $\hat{y} = \Pi.D(S_{\mathcal{S}}, y)$ in Step 5. In Step 6 and Step 7, $\mathcal{S}$ and Q invoke $\mathcal{F}_{\mathsf{mpOPRF}}$, and Q has $y' = f_{k_{\mathcal{S}}}(y)$. Q then

evaluates $r' = \hat{y} - y'$. Finally, $\mathcal{S}$ and Q invoke $\mathcal{F}_{\mathsf{EQ}}$ to test whether $r = r'$.

We note that, supposing $u \in X$ and $u = x_{i^*}$ for some i^*, we have that

$$y = \Pi.D(S_U, u) = \Pi.D(S_U, x_{i^*}) = y_{i^*},$$

$$\hat{y} = \Pi.D(S_{\mathcal{S}}, y) = \Pi.D(S_{\mathcal{S}}, y_{i^*}) = f_{k_{\mathcal{S}}}(y_{i^*}) + r \bmod 2^{\lambda}.$$

So,

$$r' = \hat{y} - y' = f_{k_{\mathcal{S}}}(y_{i^*}) + r \bmod 2^{\lambda} - f_{k_{\mathcal{S}}}(y_{i^*}) = r$$

if $u \in X$. Therefore, if $u \in X$, $\mathcal{S}$ and Q will secret-share the result that $r' = r$ in the invocation of $\mathcal{F}_{\mathsf{EQ}}$.

In contrast, supposing that $u \notin X$, the decoding result $\hat{y}$ will not equal $f_{k_{\mathcal{S}}}(y_{i^*}) + r$ (with overwhelming probability). Thus, the resulting r' is not equal to r, and $\mathcal{S}$ and Q will secret-share the result that $r' \neq r$ in the invocation of $\mathcal{F}_{\mathsf{EQ}}$.

4.3 Protocol Analysis

We first analyze the communication and computation complexity of the Π_{3PPSMT} protocol in Theorem 2 and 3.

Theorem 2. *Supposing that the OKVS scheme Π has an expansion ratio $1 + \varepsilon$, the communication complexity of the Π_{3PPSMT} protocol is $O(\lambda|X|)$ in the preprocessing phase and $O((1 + \varepsilon)\lambda|X|)$ in the online phase.*

Proof. In the preprocessing phase, the set provider U sends $f_{k_U}(x_i)$ to the server $\mathcal{S}$ for all $x_i \in X$. For each $f_{k_U}(x_i)$, it requires $O(\lambda)$ communication complexity to send it. Thus, the preprocessing phase has a total communication complexity of $O(\lambda|X|)$.

In the online phase, U sends an OKVS encoding S_U to Q, and $\mathcal{S}$ sends an OKVS encoding $S_{\mathcal{S}}$ to Q in Step 4. Each OKVS encoding stores $|X|$ key-value pairs, where each value is in the space $\{0,1\}^{\lambda}$. Since the OKVS scheme Π has an expansion ratio $1 + \varepsilon$, the encodings' size are both $(1 + \varepsilon)\lambda|X|$. Then, in the invocation of $\mathcal{F}_{\mathsf{mpOPRF}}$ and $\mathcal{F}_{\mathsf{EQ}}$, the communication complexity is $O(\lambda)$. Therefore, the communication complexity of the online phase is $O((1+\varepsilon)\lambda|X|)$. □

Theorem 3. *Supposing that the OKVS scheme has an expansion ratio $1 + \varepsilon$, the computation complexity of the set provider U is $O((1 + \varepsilon)\lambda|X|)$ in the preprocessing phase (server and querier have no computation in the preprocessing phase), and $O((1 + \varepsilon)\lambda|X|)$ for both the server and the querier in the online phase.*

Proof. The computation complexity in the preprocessing phase involves:

- The set provider U evaluates $f_{k_U}(x_i)$ for all $x_i \in X$, which has $O(\lambda|X|)$ computation complexity.
- The OKVS encoding operation stores $|X|$ key-value pairs, which has $O((1 + \varepsilon)\lambda|X|)$ computation complexity.

Therefore, the preprocessing phase of the Π_{3PPSMT} protocol has $O((1 + \varepsilon)\lambda|X|)$ computation complexity for the set provider.

The computation complexity in the online phase involves:

- The OKVS encoding operation stores $|X|$ key-value pairs in Step 3, which has $O((1 + \varepsilon)\lambda|X|)$ computation complexity for the server.
- The OKVS decoding operations have $O((1 + \varepsilon)\lambda|X|)$ computation complexity for the querier.
- The invocation of $\mathcal{F}_{\mathsf{mpOPRF}}$ and $\mathcal{F}_{\mathsf{EQ}}$ has $O(\lambda)$ computation complexity for both the server and the querier.

Therefore, the online phase of the Π_{3PPSMT} protocol has $O((1 + \varepsilon)\lambda|X|)$ computation complexity for both the server and the querier. □

In the following, we analyze the privacy of the Π_{3PPSMT} protocol. Theorem 4 states that the protocol securely implements the $\mathcal{F}_{\mathsf{3PPSMT}}$ functionality in the semi-honest threat model. In the theorem, we prove that the querier Q's query u is revealed to the server $\mathcal{S}$ and the set provider U, and no knowledge on U's set X besides its size is revealed to $\mathcal{S}$ and Q.

Theorem 4. *The Π_{3PPSMT} protocol presented in Algorithm 1 securely implements the $\mathcal{F}_{3PPSMT}$ functionality against semi-honest polynomial-time adversaries, given that $\mathcal{S}$, U, and Q are non-collusive.*

Proof. We prove this theorem by proving the following statements.

(1) $\mathcal{S}$ cannot extract any additional knowledge on Q's query u and U's set X besides $b_{\mathcal{S}}$ and $|X|$.
To prove this statement, we note that y_i received by $\mathcal{S}$ in Step 2 is computationally indistinguishable from random number due to the property of pseudorandom functions. Thus, $\mathcal{S}$ cannot extract additional knowledge from the $|X|$ random values.

(2) Q cannot extract any additional knowledge on U's set X besides b_Q and $|X|$.
To prove this statement, we first note that y_i received in Step 2 and $f_{k_{\mathcal{S}}}(y_i)$ received in Step 3 are computationally indistinguishable from random numbers due to the property of pseudorandom functions. Then, we can also observe that x_i in Step 2 and y_i in Step 3 are also computationally indistinguishable from random numbers due to the obliviousness of the OKVS scheme by Definition 7. Additionally, y' in Step 7 are also computationally indistinguishable from random numbers due to the property of pseudorandom functions.
In summary, messages received by Q can be seen as two OKVS encodings in Step 4, which store $|X|$ key-value pairs with random keys and random values, a random number in Step 7, and b_Q from $\mathcal{F}_{\mathsf{EQ}}$ in Step 8. This view can be easily simulated given $|X|$ and b_Q.

(3) U cannot extract any knowledge.
This statement can be proven by observing that U does not receive any message in the Π_{3PPSMT} protocol.

The above analysis shows that the Π_{3PPSMT} protocol in Algorithm 1 securely implements the $\mathcal{F}_{\mathsf{3PPSMT}}$ functionality. □

5 THE SECTRIC PROTOCOL

5.1 Intuitive Construction

We begin by presenting an intuitive construction of the Sectric protocol. This construction is termed "intuitive" because, while our overarching privacy objective is to ensure neighbor privacy for all graph nodes, the proposed construction guarantees neighbor privacy for every node except the querier Q. In particular, it discloses the identities of the querier's neighbors to the server.

Algorithm 2 An intuitive construction of Sectric.

Parameters: The security parameter λ and degree upper bound $D \geq \max_{u \in \mathcal{V}} |N_u|$.
Involved parties: The server $\mathcal{S}$ and the graph nodes $\mathcal{V}$.
Private inputs: N_u for $u \in \mathcal{V}$.
Cryptographic primitive: The $\mathcal{F}_{\text{3PPSMT}}$ functionality.
Protocol contents:

1: Denote the querier in $\mathcal{V}$ as Q.
2: **for** $u \in \mathcal{V}$ and $|N_u| < D$ **do** u uniformly samples x from $\{0,1\}^\lambda$. u adds x into N_u.
3: **end for**
4: **for** $u \in N_Q, v \in N_Q/\{u\}$ **do**
5: $\mathcal{S}$ acts as the server with input $\perp$.
6: Q acts as the querier with input v.
7: u acts as the set provider with input N_u.
8: $\mathcal{S}$, Q, and u invoke the protocol $\mathcal{F}_{\text{3PPSMT}}$.
9: Let $\mathcal{S}$ obtain $b_{\mathcal{S}}^{u,v}$ from $\mathcal{F}_{\text{3PPSMT}}$.
10: Let Q obtain $b_{Q}^{u,v}$ from $\mathcal{F}_{\text{3PPSMT}}$.
11: **end for**
12: $\mathcal{S}$ evaluates $b_{\mathcal{S}} = \sum b_{\mathcal{S}}^{u,v} \pmod{2^\lambda}$.
13: Q evaluates $b_Q = \sum b_{Q}^{u,v} \pmod{2^\lambda}$.
14: $\mathcal{S}$ sends $b_{\mathcal{S}}$ to Q.
15: Q calculates $|\Delta_Q| = b_{\mathcal{S}} - b_Q \pmod{2^\lambda}$.

We introduce this intuitive construction because it captures the core idea of our full protocol and is easy to understand. Presenting this simplified version can help readers grasp the essential concepts before we describe the complete secure protocol.

Following the description of the intuitive protocol, we will elaborate on why it preserves neighbor privacy for all nodes other than the querier, as well as the reasons for its inability to protect the identities of the querier's neighbors.

Protocol description. Sectric is executed on a decentralized graph. The node set is public, and the edge set is distributed among all graph nodes, where each graph node has its neighbor set. The protocol involves the server $\mathcal{S}$ and graph nodes $\mathcal{V}$ as the protocol participants. The server $\mathcal{S}$ has no input and each graph node $u \in \mathcal{V}$ has its neighbor set N_u as the input.

Suppose that a graph node $Q \in \mathcal{V}$ initiates a local triangle counting task and queries the number of its local triangles. To fulfill this task, $\mathcal{S}$ and Q first target at secret-sharing $|N_Q \cap N_u|$ for each $u \in N_Q$. To achieve this, they test for each $v \in N_Q$ whether $v \in N_u$ and secret-share the result. This task can be done through invoking the 3PPSMT functionality $\mathcal{F}_{\text{3PPSMT}}$.

Therefore, the intuitive protocol works as follows. Given a neighbor $u \in N_Q$, $\mathcal{S}$, Q, and u invoke the 3PPSMT functionality $\mathcal{F}_{\text{3PPSMT}}$. They act as the server, the querier, and the set provider, respectively, for each $v \in N_Q$, where Q and u use v and N_u as the inputs. In each invocation, $\mathcal{S}$ and Q obtain the outputs. This step tests whether v is in N_u for all $v \in N_Q$, and $\mathcal{S}$ and Q share the result. Aggregating the secret-shares of the results, Q and $\mathcal{S}$ secret-share $|N_Q \cap N_u|$ for this neighbor u. $\mathcal{S}$ and Q repeat the above procedure for all $u \in N_Q$ and secret-share $|N_Q \cap N_u|$ for all $u \in N_Q$. Finally, $\mathcal{S}$ and Q sum the secret-shares, respectively. By Theorem 1, they obtain a secret share of

$$\frac{1}{2} \sum_{u \in N_Q} |N_Q \cap N_u| = |\Delta_Q|.$$

$\mathcal{S}$ sends its share to Q, and then Q can recover the result $|\Delta_Q|$.

Privacy for querier's neighbors. We first briefly analyze why this intuitive construction provides privacy to the querier's neighbors. Suppose that the $\mathcal{F}_{\text{3PPSMT}}$ functionality is securely implemented by a protocol Π_{3PPSMT} in the universal composability model. We note that messages received by $\mathcal{S}$ in this protocol is identical to these in the instances of Π_{3PPSMT}. Due to the property of universal composability, the parallel composition of multiple Π_{3PPSMT} instances is also secure. So, no knowledge of the participants' neighbor sets is revealed to $\mathcal{S}$. This fact also holds for Q's neighbors in N_Q. For Q, the situation is somewhat different. Messages received by Q consists of these in Π_{3PPSMT} instances and the secret-share $b_{\mathcal{S}}$ of $|\Delta_Q|$ from $\mathcal{S}$. This does not alter the protocol's privacy requirement, as $b_{\mathcal{S}}$ is obtained from b_Q and $|\Delta_Q|$, which are both known to Q.

Insecurity of the intuitive protocol. The reason that this intuitive construction reveals the querier's neighbors lies in the fact that the server $\mathcal{S}$ can tell Q's neighbors by observing which graph nodes it interacts with in the instances of Π_{3PPSMT}.

In Section 5.2, we provide a technique to securely compose multiple instances of the 3PPSMT protocol to fix the above insecurity.

5.2 Full Protocol from Secure Composition

In the following, we describe the full construction of the protocol. This construction securely fulfills the privacy-preserving local triangle counting task.

Recall that multiple instances of the Π_{3PPSMT} protocol are invoked in Step 2-6 in the intuitive construction in Algorithm 2. In each instance, the querier Q specifies a node $u \in \mathcal{V}$ and an element $v \in N_Q$, and tests whether $v \in N_u$.

In this full construction, we mainly demonstrate how to securely compose these Π_{3PPSMT} protocol instances, so that the server $\mathcal{S}$ does not interact with Q's neighbors. Integrating this secure composition technique into the intuitive protocol in Algorithm 2, we obtain the full construction of Sectric.

We model the process as a group of set providers (i.e., the graph nodes $\mathcal{V}$) each of which provides a set (i.e., $u \in \mathcal{V}$ provides N_u). The querier has m queries, each of which specifies a set provider (i.e., $u \in N_Q$) and an element (i.e., $v \in N_Q/\{u\}$). Assisted by the server, the querier tests for each query whether the element is in the corresponding provider's set. The privacy requirement is that the set provider in each query is not revealed.

As stated before, naively composing multiple Π_{3PPSMT} instances in parallel reveals the set providers specified by Q to $\mathcal{S}$ because the server has to interact with them. In the following, we describe how the secure composition technique fixes such leakage. Corresponding to the original Π_{3PPSMT} protocol, we describe the composition of the preprocessing phase and the online phase, respectively.

Composition of the preprocessing phase. In order to compose the preprocessing phase, we first prove Theorem 5, which states that the preprocessing phase of the Π_{3PPSMT} protocol is reusable.

Algorithm 3 Secure composition of multiple Π_{3PPSMT} protocol instances.

Public parameters:

- The security parameter λ.
- The query size m and table size γm ($\gamma > 1$).

Involved parties: The server $\mathcal{S}$, the querier Q, and a group of set providers $\mathcal{V}$.

Private inputs:

- For $u \in \mathcal{V}$: $N_u \subset \{0,1\}^\lambda$.
- For Q: m queries $\{(v_i, x_i)\}_{i\in[m]} \subset \mathcal{V} \times \{0,1\}^\lambda$.

Cryptographic primitives: Same as the Π_{3PPSMT} protocol.

Preprocessing:

1: **for** $u \in \mathcal{V}$ **do**
2: u uniformly samples $k_u \xleftarrow{\$} \{0,1\}^\lambda$ and evaluates $y_{u,i} = f_{k_u}(x_i)$ for all $x_i \in N_u$.
3: u encodes $S_u \leftarrow \Pi.E(\langle (x_i, y_{u,i})\rangle_{x_i \in N_u})$, and sends all $\langle y_{u,i}\rangle_{x_i\in N_u}$ to $\mathcal{S}$.
4: **end for**
5: $\mathcal{S}$ builds a table T of γm entries.
6: For all $u \in \mathcal{V}$ and $x_i \in N_u$, $\mathcal{S}$ stores $y_{u,i}$ in the ind_1-th, ind_2-th, and ind_3-th entries of T, where $ind_j = \mathcal{H}_j(y_{u,i})$ for $j \in \{1,2,3\}$.

Online phase:

7: $\mathcal{S}$ samples $k_{\mathcal{S}} \xleftarrow{\$} \{0,1\}^\lambda$.
8: **for** $j \in [\gamma m]$ **do**
9: Let $y_1, \ldots, y_{l_j}$ denote the strings stored in the j-th entry of T.
10: $\mathcal{S}$ samples $r_j \xleftarrow{\$} \{0,1\}^\lambda$ and encodes $S_{\mathcal{S},j} = \Pi.E(\langle (y_i, f_{k_{\mathcal{S}}}(y_i) + r_j \mod 2^\lambda)\rangle_{i\in[l_j]})$.
11: **end for**
12: $\mathcal{S}$ sends all $S_{\mathcal{S},j}$ to Q. Q asks v_i for S_{v_i} for all v_i.
13: For each query (v_i, x_i), Q decodes $\tilde{y}_i \leftarrow \Pi.D(S_{v_i}, x_i)$.
14: Q builds a cuckoo hash table of γm entries storing all $\tilde{y}_i$ using $\mathcal{H}_1, \mathcal{H}_2$, and $\mathcal{H}_3$.
15: Let y_j be the j-th entry of the cuckoo hash table if the entry is not empty, and otherwise let $y_j = 0$. Q evaluates $\hat{y}_j \leftarrow \Pi.D(S_{\mathcal{S},j}, y_j)$.
16: $\mathcal{S}$ and Q invoke $\mathcal{F}_{\mathsf{mpOPRF}}$ as the sender and the receiver with inputs $k_{\mathcal{S}}$ and $\langle y_j\rangle_{j\in m}$, respectively.
17: Supposing Q obtains $\langle y'_j\rangle_{j\in m}$ from $\mathcal{F}_{\mathsf{mpOPRF}}$, it evaluates $r'_j = \hat{y}_j - y'_j \mod 2^\lambda$.
18: For $j \in [\gamma m]$, Q and $\mathcal{S}$ invoke $\mathcal{F}_{\mathsf{EQ}}$ with inputs r_j and r'_j, respectively, and have outputs $b_{\mathcal{S},j}$ and $b_{Q,j}$. They evaluate $b_{\mathcal{S}} = \sum b_{\mathcal{S},j} \mod 2^\lambda$ and $b_Q = \sum b_{Q,j} \mod 2^\lambda$ as the protocol output.

Theorem 5. *The preprocessing phase of the Π_{3PPSMT} protocol is reusable. In other words, the protocol supports polynomially many independent queries after a single preprocessing.*

Proof. From the protocol construction, messages received by the server and the querier in multiple invocations of the online phase are independent given that the randomness r is newly selected in each invocation. Thus, they cannot obtain more information on the set provider's set in parallel invocations of the Π_{3PPSMT} protocol due to the universal composability of the $\mathcal{F}_{\mathsf{mpOPRF}}$ and $\mathcal{F}_{\mathsf{EQ}}$ functionalities. Therefore, the preprocessing phase of the protocol is reusable. □

This theorem inspires the composition of the preprocessing phase. Specifically, we let all set providers run the preprocessing phase of Π_{3PPSMT} in parallel and send preprocessing results to the server $\mathcal{S}$. With these preprocessing results, the server no longer has to interact with the set providers in the online phase.

Composition of the online phase. Then, we consider how to compose the online phase with the preprocessing results. As the server has possessed all preprocessing results, it can respond to the querier using the preprocessing result corresponding to the queried set provider. However, the querier still has to let the server know the queried set provider.

To fix this problem, we have a key observation that the querier Q knows the queried set providers $\{v_i\}_{i\in[m]}$. Thus, Q can ask v_i for $S_{v_i} \leftarrow \Pi.E(\langle (x', f_{k_u}(x'))\rangle_{x'\in N_{v_i}})$ as Step 4 in Algorithm 1. After obtaining these OKVS encodings, Q decodes S_{v_i} on the queried element x_i for each query (v_i, x_i). The decoding result $\Pi.D(S_{v_i}, x_i) = f_{k_{v_i}}(x_i)$ supposing $x_i \in N_{v_i}$, and otherwise is a pseudorandom number.

With the decoding results, the problem is reduced to how to test whether each decoding result $\Pi.D(S_{v_i}, x_i)$ is in the PRF values $\langle f_{k_{v_i}}(x)\rangle_{x\in N_{v_i}}$ sent by the corresponding set provider v_i. The key challenge is how to specify $\langle f_{k_{v_i}}(x)\rangle_{x\in N_{v_i}}$ among all the PRF values received by the server without revealing the exact identity of v_i to the server.

We address this challenge by observing that, due to the pseudorandomness of PRF values, the PRF values from different set providers make a collision with only negligible probability. Thus, the querier only has to test whether each decoding result is in all the PRF values possessed by the server. This task can be directly fulfilled using the technique from the Π_{3PPSMT} protocol's online phase in Algorithm 1.

Overheads optimization. In the preprocessing phase of the above solution, the server $\mathcal{S}$ totally receives $O(\sum_{u\in\mathcal{V}} |N_u|) = O(|\mathcal{E}|)$ PRF values. The communication overhead is $O((1+\varepsilon)\lambda|\mathcal{E}|)$ for each query, and totally $O((1+\varepsilon)m\lambda|\mathcal{E}|)$ in the online phase. Similarly, the total computation overhead is $O((1+\varepsilon)m\lambda|\mathcal{E}|)$.

To further optimize the overheads, we integrate the technique of cuckoo hash. The decoding results are mapped to different entries of the querier-side cuckoo hash table, and the PRF values are mapped to all possible entries in the server-side table. Then, the membership test is only applied to the decoding result and the PRF values in the same entry of querier-side and server-side tables.

Through this method, we reduce the online-phase communication overheads from $((1+\varepsilon)m\lambda|\mathcal{E}|)$ to $O((1+\varepsilon)\lambda|\mathcal{E}|)$, and reduce the computation overheads from $O((1+\varepsilon)m\lambda|\mathcal{E}|)$ to $O((1+\varepsilon)\lambda|\mathcal{E}|)$ and $O((1+\varepsilon)\lambda\gamma m)$ for the server and the querier, respectively. More details can be found in Theorem 6 and Theorem 7.

Combining the above techniques, the protocol implementing the secure composition of multiple Π_{3PPSMT} instances is presented in

Algorithm 3. Integrating this protocol into the intuitive construction of Sectric in Algorithm 2 to replace Step 3-6, we obtain the full construction of Sectric.

5.3 Theoretical Analysis

In the following, we analyze the proposed protocols from a theoretical perspective. We first demonstrate the communication and computation cost of invoking multiple instances of Π_{3PPSMT} using the secure composition technique proposed in Algorithm 3.

Theorem 6. *Given that the OKVS scheme has an expansion ratio $1+\varepsilon$, the communication complexity of the composed protocol in Algorithm 3 is $O(\lambda|\mathcal{E}|)$ in the preprocessing phase and $O((1+\varepsilon)\lambda|\mathcal{E}|)$ in the online phase.*

Proof. In the preprocessing phase, the communication mainly occurs in Step 3, where each $u \in \mathcal{V}$ sends all PRF values $y_{u,i}$ to $\mathcal{S}$. There are a total of $\sum_{u\in\mathcal{V}} |N_u| = |\mathcal{E}|$ PRF values of size $O(\lambda)$. Thus, the communication complexity of the preprocessing phase is $O(\lambda|\mathcal{E}|)$.

In the online phase, the communication mainly occurs in Step 12, where $\mathcal{S}$ sends all $S_{\mathcal{S},j}$ to Q, and for each query (v_i, x_i), v_i sends S_{v_i} to Q. In the former, $S_{\mathcal{S},j}$ stores l_j key-value pairs with value size $O(\lambda)$. So, $S_{\mathcal{S},j}$ has size $(1+\varepsilon)\lambda l_j$. Noting that all entries of T store $O(\sum_{u\in\mathcal{V}} |N_u|) = O(|\mathcal{E}|)$ PRF values, these OKVS encodings have a total size of $O(\sum(1+\varepsilon)\lambda l_j) = O((1+\varepsilon)\lambda|\mathcal{E}|)$. In the latter, each OKVS encoding S_{v_i} stores $|N_{v_i}|$ key-value pairs with value size $O(\lambda)$. So, S_{v_i} has size $O((1+\varepsilon)\lambda|N_{v_i}|)$. Noting that for repeated v_i, Q only has to ask for S_{v_i} once. Thus, it requires $O(\sum_{u\in\mathcal{V}} S_u) = O(\sum_{u\in\mathcal{V}}(1+\varepsilon)\lambda|N_u|) = O((1+\varepsilon)\lambda|\mathcal{E}|)$ communication complexity to send S_{v_i} for all queries (v_i, x_i). Therefore, the communication complexity in the online phase is $O((1+\varepsilon)\lambda|\mathcal{E}|)$. □

Theorem 7. *Assuming the OKVS scheme has an expansion ratio $1+\varepsilon$, the computation complexity of the secure composition technique in Algorithm 3 is $O((1+\varepsilon)\lambda|N_u|)$ for graph node $u \in \mathcal{V}$ and $O(\lambda|\mathcal{E}|)$ for the server $\mathcal{S}$ in the preprocessing phase. In the online phase, the computation complexity is $O((1+\varepsilon)\lambda|\mathcal{E}|)$ for the server $\mathcal{S}$ and $O((1+\varepsilon)\lambda\gamma m)$ for the querier Q.*

Proof. In the preprocessing phase, each graph node $u \in \mathcal{V}$ evaluates all $f_{k_u}(x_i)$ for all $x_i \in N_u$, requiring $O(\lambda|N_u|)$ computation complexity, and stores $|N_u|$ key-value pairs with value size $O(\lambda)$ in the OKVS encoding S_u, requiring $O((1+\varepsilon)\lambda|N_u|)$ computation cost. Thus, the computation complexity for graph node u is $O((1+\varepsilon)\lambda|N_u|)$.

The server $\mathcal{S}$ has $O(\sum_{u\in\mathcal{V}} \lambda|N_u|) = O(\lambda|\mathcal{E}|)$ computation cost to build the table T. Thus, the computation complexity for graph node u is $O(\lambda|\mathcal{E}|)$.

In the online phase, the server $\mathcal{S}$ evaluates γm OKVS encodings, each of which stores l_j key-value pairs with pair size $O(\lambda)$. So, its computation complexity is $O(\sum(1+\varepsilon)l_j\lambda) = O((1+\varepsilon)\lambda|\mathcal{E}|)$. $\mathcal{S}$ also invokes γm instances of $\mathcal{F}_{\text{mpOPRF}}$ and $\mathcal{F}_{\text{EQ}}$, making $O(\gamma\lambda m)$ computation complexity. Noting that $O(\gamma m) \le O((1+\varepsilon)|\mathcal{E}|)$, the computation complexity of the server is $O((1+\varepsilon)\lambda|\mathcal{E}|)$.

The computation of the querier Q mainly occurs in Step 13 and 15, decoding γm OKVS encodings with value size λ. This makes $O((1+\varepsilon)\lambda\gamma m)$ computation complexity. □

Integrating the secure composition technique presented in Algorithm 3 into the intuitive construction of Sectric in Algorithm 2 to replace Step 3-6, we obtain the full construction of Sectric. In the following, we analyze the privacy guarantee of this full construction of Sectric, which is demonstrated in Theorem 8.

Theorem 8. *Sectric satisfies the privacy constraint proposed in Definition 3.*

Proof. From the description of the protocol, the views of the server and the querier in Sectric are the same as the joint view in multiple invocations of the Π_{3PPSMT} protocol. So, by Theorem 4 and Theorem 5, only $|N_u|$ for each u is revealed to $\mathcal{S}$, and $\sum_{u\in\mathcal{V}} |N_u|$ is revealed to Q. As $|N_u| = D$ for all $u \in \mathcal{V}$ in the invocations, this does not reveal the privacy of protocol participants. □

As the main body of Sectric invokes multiple instances of the Π_{3PPSMT} protocol using the secure composition technique, we do not analyze its overheads for brevity.

5.4 DP Extension of Sectric

In the following, we propose DPSectric, an extension of Sectric that outputs an estimated result satisfying the edge-level central differential privacy (CDP) mechanism. Quite surprisingly, we find that only a simple modification is sufficient to achieve this goal. Specifically, we extend Sectric as follows to provide edge-level CDP: in Step 14 of Algorithm 2, $\mathcal{S}$ sends $b'_{\mathcal{S}} = b_{\mathcal{S}} + \text{Lap}(\frac{1}{\epsilon})$ instead of $b_{\mathcal{S}}$ to the querier Q, where $\text{Lap}(\cdot)$ denotes the Laplace distribution and ϵ is the privacy budget.

Next, we analyze how this extension provides edge-level CDP. We note that before the server $\mathcal{S}$ sends its share $b_{\mathcal{S}}$ to the querier Q in the final step of Algorithm 2, the querier has no knowledge about the adjacency relations of other graph nodes. Thus, the server can add noise to its share to ensure edge-level CDP.

The next question is how much noise should be added. This depends on the sensitivity of local triangle counts, defined as the maximum change in the number of local triangles involving a node between two graphs differing by a single edge. Since the querier already knows all edges incident to itself, we only need to consider changes to edges that do not involve the querier. In this case, the difference in local triangle counts is at most one. Therefore, the sensitivity is 1.

Based on the above analysis, we formally prove the utility and privacy guarantees of DPSectric.

Utility. To demonstrate the utility of this DPSectric, we prove that the result is unbiased and analyze the l_2 loss of the result.

Theorem 9. *Let $f(G,Q)$ be the number of local triangles containing Q in G, and $\hat{f}(G,Q)$ be the estimate of Sectric with this extension. Then, we have $\mathbb{E}(\hat{f}(G,Q)) = f(G,Q)$ (i.e. the estimate is unbiased), and*

$$l_2^2(f(G,Q), \hat{f}(G,Q)) \le \frac{2}{\epsilon^2}.$$

Proof. In the extension, $\mathcal{S}$ sends $b'_{\mathcal{S}} = b_{\mathcal{S}} + \text{Lap}(\frac{1}{\epsilon})$ to Q, and Q outputs the estimate $\hat{f}(G,Q) = b'_{\mathcal{S}} - b_Q = b_{\mathcal{S}} + \text{Lap}(\frac{1}{\epsilon}) - b_Q = f(G,Q) + \text{Lap}(\frac{1}{\epsilon})$. Thus, $\mathbb{E}(\hat{f}(G,Q)) = \mathbb{E}(f(G,Q) + \text{Lap}(\frac{1}{\epsilon})) = f(G,Q)$. For the l_2 loss, we have

$$l_2^2(f(G,Q),\hat{f}(G,Q)) = l_2^2(f(G,Q), f(G,Q) + \text{Lap}(\frac{1}{\epsilon})) = l_2^2(0, \text{Lap}(\frac{1}{\epsilon})) \leq \frac{2}{\epsilon^2}.$$

□

Privacy. Then, we prove that DPSectric provides ϵ-Edge CDP.

Theorem 10. *The output DPSectric satisfies the ϵ-Edge CDP proposed in Definition 4 against a polynomial-time adversary.*

Proof. As stated above, we only need to consider two graphs differing in one edge which does not contain the querier, which implies the difference of local triangle counts is at most one. Let $\hat{L}(G)$ and $\hat{L}(G')$ denote the local triangle counts computed by DPSectric, while $L(G)$ and $L(G')$ represent those computed by Sectric. Additionally, $S(G)$ and $S(G')$ indicate the server's share, and $Q(G)$ and $Q(G')$ denote the querier's share. We could know that $L(G) = S(G) - Q(G)$ and $L(G') = S(G') - Q(G')$. Define $|L(G) - L(G')| = \Delta_l$, where Δ_l is the sensitivity of the local triangle count. The noises Y and Y' are drawn from $\text{Lap}(\frac{\Delta_l}{\epsilon})$. The probability of outputting the same local triangle counting result is

$$\frac{\Pr[\hat{L}(G) = l]}{\Pr[\hat{L}(G') = l]} = \frac{\Pr[S(G) + Y - Q(G) = l]}{\Pr[S(G') + Y' - Q(G') = l]} = \frac{\Pr[L(G) + Y = l]}{\Pr[L(G') + Y' = l]} = \frac{\Pr[Y = l - L(G)]}{\Pr[Y' = l - L(G')]} = \frac{e^{\frac{\epsilon|l-L(G)|}{\Delta_l}}}{e^{\frac{\epsilon|l-L(G')|}{\Delta_l}}} = e^{\frac{\epsilon(|l-L(G)|-|l-L(G')|)}{\Delta_l}} \leq e^{\frac{\epsilon|L(G)-L(G')|}{\Delta_l}} = e^{\epsilon}$$

□

6 IMPLEMENTATION AND EVALUATION

We perform extensive experiments on the performance of Sectric and present the results in this section.

6.1 Experiment Setting

Datasets. The experiments are conducted on both real-world and synthetic graphs for comprehensiveness. The real-world graphs are collected from SNAP [27] and Network Repository [44], including Facebook, CondMat, roadNet, email-Enron, and loc-Brightkite. Details of the real-world graphs are provided in Table 3, where $|\mathcal{V}|$ represents the number of nodes, $|\mathcal{E}|$ denotes the number of edges, $d_{\max}$ indicates the maximum degree, and *Domain* denotes the types of graphs. The synthetic graphs are generated using the Problem Based Benchmark Suite (PBBS) [48] under different parameters.

Implementation Details. We implement our experimental evaluations using C++ programming. All our experiments are conducted on an Ubuntu virtual machine configured with a 2.6 GHz Intel i9-13900H, equipped with 4 cores and 32 GB of RAM. Unless otherwise specified, we record the computation time at a network speed of 10 Gbps and measure the online communication overhead as the amount of data transferred between the server and the querier. Our implementation targets an error probability of 2^{-40} and 128 bits of computational security, assuming that the server and the querier have pre-generated the Beaver Triples in advance. We integrate the OKVS and OPRF [42] implementations from [65], as well as the private equality test implementation from [46].

Table 3: Details of Real-World Graph Datasets.

Graph	*Abbr.*	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$d_{\max}$	*Domain*
Facebook	FB	4,039	88,234	1,045	Social Network
CondMat	CM	23,133	93,497	279	Collaboration Network
roadNet	RN	1,379,917	165,435	7	Road Network
email-Enron	EE	36,692	183,831	1,383	Communication Network
loc-Brightkite	LB	58,228	214,078	1,134	Social Network

Table 4: Online-Phase Overheads on Real-world Graphs.

Graph	Communication (GB)				Computation (s)		
	CARGO	Sectric			CARGO	Sectric	
		Q → S	S → Q	Total		Querier	Server
FB	**0.0911**	0.7341	0.8924	1.6265	**2.1774**	46.5642	46.3558
CM	2.9899	0.0543	0.4137	**0.4679**	70.4530	8.3097	**8.1869**
RN	-	0.0019	1.2405	**1.2423**	-	52.4064	**52.2829**
EE	7.5224	1.2241	3.9794	**5.2035**	**141.7760**	185.0280	184.9030
LB	18.9449	0.8500	4.4225	**5.2725**	443.8780	248.4930	**248.3810**

The runtime of CARGO on roadNet exceeds one hour. We use "-" to represent this in the table.

Table 5: Preprocessing Overheads on Real-world Graphs.

Graph	Communication		Computation	
	Node (KB)	Server (MB)	Node (ms)	Server (s)
FB	16.33	64.40	1.1	5.14
CM	4.36	98.48	1.0	6.01
RN	0.19	336.89	0.7	14.60
EE	21.61	774.31	74.1	48.91
LB	17.72	1007.54	1.1	75.26

Baselines. Sectric is the first crypto-assisted solution specifically designed for privacy-preserving local triangle counting. To set a proper performance baseline, we adopt CARGO [35], the state-of-the-art crypto-assisted approach for global triangle counting. To enable a fair comparison, we modify its open-source implementation [51] to return local triangle counts in our experiments[2].

6.2 Experiment Results on Real-world Graphs

We first conduct experiments on real-world graphs to evaluate the practical performance of Sectric.

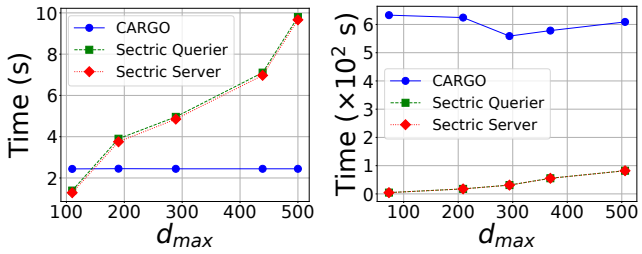
Online-Phase Overheads. To demonstrate the efficiency of our solution, we compare the online-phase overheads of Sectric with those of the baseline, and present the results in Table 4. The experiment results show that Sectric outperforms on large-scale graphs.

[2]In the original version of CARGO, it introduces two non-colluding servers to secret-share the adjacency table. The two servers compute the number of local triangles for each node and secret-share the results, which are then aggregated to obtain the total number of triangles in the entire graph. In our modification, we remove the final aggregation step to directly return the local triangle count for a given node.

Table 6: End-to-end Querying Latency on Real-world Graphs.

Graph	CARGO (s)		Sectric 1-thread (s)		Sectric 8-thread (s)	
	10 Gbps	100 Mbps	10 Gbps	100 Mbps	10 Gbps	100 Mbps
FB	**2.250**	**9.640**	46.564	176.340	27.718	155.300
CM	72.845	315.386	8.310	51.639	**4.436**	**44.331**
RN	-	-	52.406	188.384	**14.718**	**140.962**
EE	147.794	758.011	185.028	629.693	**100.569**	**542.300**
LB	459.034	1995.844	248.493	743.878	**136.280**	**581.200**

The runtime of CARGO on roadNet exceeds one hour. We use "-" to represent this in the table.



(a) $|\mathcal{V}| = 4000$ **(b)** $|\mathcal{V}| = 60000$

Figure 2: Computation overheads in the online phase on synthetic graphs with the same $|\mathcal{V}|$ and different d_{max}.

On the loc-Brightkite graph, our protocol reduces computation overheads by 44.02% and communications overheads by 72.17% in the online phase.

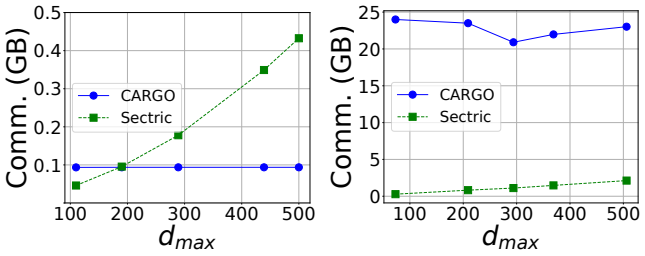
Preprocessing-Phase Overheads. We also evaluate the overheads of Sectric in the preprocessing phase on real-world graphs and present the experimental results in Table 5. Although CARGO also includes a preprocessing phase, the authors do not provide an open-source implementation. Therefore, we only report the preprocessing overheads of Sectric.

The results show that the preprocessing phase of Sectric can be completed using approximately 1 GB of memory and within two minutes. Since the preprocessing phase is executed only once, such overhead is considered acceptable.

End-to-end Overheads with Multi-thread Optimization. We also implement a multi-threaded version of Sectric and evaluate its end-to-end performance under varying levels of parallelism and network bandwidth. The experimental results are presented in Table 6. The results demonstrate that, with 8-thread parallelization, the computation overhead can be reduced by 10% to 72% on real-world graphs.

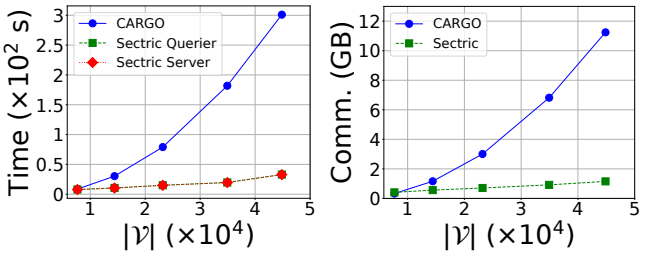
6.3 Experiment Results on Synthetic Graphs

The above experiments evaluate the practical performance of Sectric on real-world graphs. However, real-world graphs have fixed scales and structures. To further assess the performance of Sectric under varying graph characteristics, we also conduct experiments on synthetic graphs generated with different parameters. The graphs are synthesized with different maximum node degrees d_{max} and node size $|\mathcal{V}|$. In the following, we present the experimental results on these synthetic graphs.



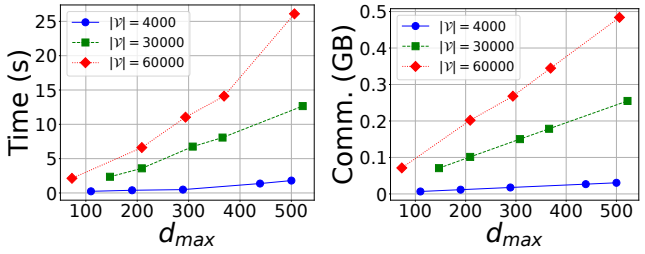
(a) $|\mathcal{V}| = 4000$ **(b)** $|\mathcal{V}| = 60000$

Figure 3: Communication overheads in the online phase on synthetic graphs with the same $|\mathcal{V}|$ and different d_{max}.



(a) Computation Overheads **(b) Communication Overheads**

Figure 4: Computation and communication overheads in the online phase on synthetic graphs with the same d_{max} and different $|\mathcal{V}|$.



(a) Computation Overheads **(b) Communication Overheads**

Figure 5: Preprocessing Overheads of Sectric on Synthetic Graphs.

Online-Phase Overheads. We first evaluate the online-phase overheads of Sectric on these synthetic graphs. The computation and communication overheads, along with comparisons to the baseline, are presented in Figure 2 and 3, respectively.

The experimental results show that the computation and communication overheads of Sectric grow linearly with $d_{\max}$, while the overheads of CARGO depend solely on the number of nodes in the graph. Sectric outperforms CARGO on graphs with a larger number of nodes. On the largest graph in our experiments, Sectric reduces computation overhead by 86.52% and communication overhead by 90.76%.

In addition, we investigate the impact of the graph size, denoted by the number of nodes $|\mathcal{V}|$, on the overheads. The results are shown in Figure 4. They indicate that Sectric's overheads grow linearly with $|\mathcal{V}|$, while CARGO's overheads grow quadratically. These findings are consistent with our theoretical analysis.

Preprocessing-Phase Overheads. We also evaluate the overheads of Sectric in the preprocessing phase on synthetic graphs. The experiment results are presented in Figure 5. The results also show that Sectric overheads in the preprocessing phase also grow linearly with d_{max} and $|\mathcal{V}|$.

6.4 Utility-privacy Trade-off of DPSectric

We also conduct experiments to assess the utility–privacy trade-off of DPSectric. We additionally adopt three state-of-the-art LDP algorithms for triangle counting: ARRFull, ARROneNs, and ARRTwoNs [22], as baselines. Note that the baseline methods—ARRFull, ARROneNs, and ARRTwoNs—originally output the total number of triangles in the entire graph. We halt ARRFull, ARROneNs, and ARRTwoNs at the user end to support local triangle counting.

We select two social networks, Facebook and loc-Brightkite, as experimental datasets. On these graphs, we assess the utility of DPSectric under varying privacy budgets ϵ and compare it against the baseline methods. In the experiments, the l_2 loss of algorithm estimates serves as the utility metric, which is calculated as

$$l_2(\epsilon) = \frac{1}{N} \sum \left(\hat{f}_\Delta(G, i; \epsilon) - f_\Delta(G, i) \right)^2,$$

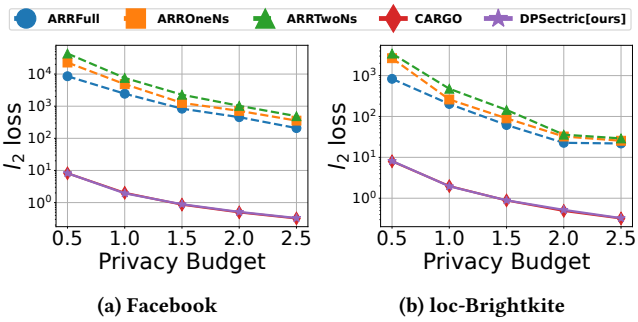
where $\hat{f}\Delta(G, i; \epsilon)$ denotes the estimate and $f\Delta(G, i)$ denotes the ground truth. The experimental results are presented in Figure 6.

The utility of DPSectric closely matches that of CARGO for that they both provide edge-level CDP and apply the Laplace mechanism, but DPSectric outperforms CARGO in efficiency on large graphs as shown in prior experiments. The results also show that DPSectric outperforms the LDP algorithms—ARRFull, ARROneNs, and ARRTwoNs—in terms of accuracy with the same privacy budget.

Besides, we demonstrate the utility contribution of DPSectric to downstream tasks with the calculation of clustering coefficient. We apply the triangle counting algorithms to calculating of clustering coefficient and assess the utility of different algorithms. The results are presented in Figure 7. The results show that when higher privacy guarantee (i.e. lower privacy budget) is required, the CDP algorithms DPSectric and CARGO achieve greater utility.

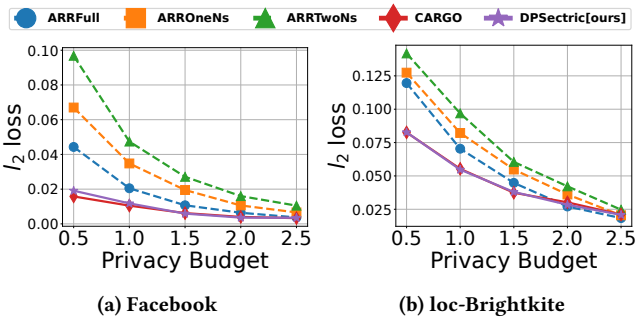
6.5 Summary of Experiments

The above experiments demonstrate that, as the overheads of Sectric grow linearly with respect to $|\mathcal{V}|$ and d_{max}, Sectric outperforms existed solutions on larger graphs and sparser graphs. From the results in Figure 4, Sectric has better performance than the baseline solution on the graph has more than 10,000 nodes with a fixed d_{max}. The experimental results in Table 6 on real-world graphs also demonstrate that Sectric incurs fewer overheads on the four larger graphs: CondMat, roadNet, email-Enron, and loc-Brightkite. Furthermore, compared to the state-of-the-art CDP and LDP methods, the DP extension of Sectric achieves higher utility with the same privacy budget. Therefore, Sectric is more suitable for the analysis of local triangle counting on large graphs.



(a) Facebook (b) loc-Brightkite

Figure 6: l_2 loss for counting local triangles with different protocols. The results for CARGO and DPSectric are overlapped.



(a) Facebook (b) loc-Brightkite

Figure 7: l_2 loss for calculating local clustering coefficient with different protocols.

7 CONCLUSION

In this work, we present Sectric, a novel server-aided crypto-assisted local triangle counting protocol. Sectric achieves both high accuracy of results and cryptographic-level privacy guarantees utilizing cryptographic primitives. It explores a novel PSI cardinality-based approach to local triangle counting with high efficiency. To avoid intermediate privacy cost, we also define and implement a new cryptographic primitive named as 3PPSMT protocol. In addition, we propose the DPSectric protocol to avoid privacy leak in the result of Sectric. We demonstrate the security of the proposed solutions through thorough theoretical analysis and evaluate the performance of the protocols through empirical experiments.

ACKNOWLEDGMENTS

This work was supported by the Natural Science Foundation on Frontier Leading Technology Basic Research Project of Jiangsu under Grant BK20222001, the National Natural Science Foundation of China under Grants No.62272222, No.62272215, No.62325205 and No.62172204, the Fundamental Research Funds for the Central Universities (No. 2024300401) the Key Program of the Natural Science Foundation of Jiangsu Province under Grant No. BK20243053, and the Nanjing University-China Mobile Communications Group Co., Ltd. Joint Institute. Sheng Zhong and Yuan Zhang are the corresponding authors.

REFERENCES

[1] Mohammad Al Hasan and Vachik S Dave. 2018. Triangle counting in large networks: a review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8, 2 (2018), e1226.

[2] Toshinori Araki, Jun Furukawa, Kazuma Ohara, Benny Pinkas, Hanan Rosemarin, and Hikaru Tsuchida. 2021. Secure Graph Analysis at Scale. *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (2021).

[3] Lars Backstrom, Cynthia Dwork, and Jon M. Kleinberg. 2007. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *The Web Conference*.

[4] Johes Bater, Yongjoo Park, Xi He, Xiao Wang, and Jennie Rogers. 2020. Saqe: practical privacy-preserving approximate query processing for data federations. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2691–2705.

[5] Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis. 2008. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 16–24.

[6] Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis. 2010. Efficient algorithms for large-scale local triangle counting. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 4, 3 (2010), 1–28.

[7] Paul Burkhardt. 2024. Triangle centrality. *ACM Transactions on Knowledge Discovery from Data* 18, 9 (2024), 1–34.

[8] Javad Ghareh Chamani, Ioannis Demertzis, Dimitrios Papadopoulos, Charalampos Papamanthou, and Rasool Jalili. 2023. GraphOS: Towards Oblivious Graph Processing. *Proceedings of the VLDB Endowment* 16, 13 (2023), 4324–4338.

[9] Nishanth Chandran, Divya Gupta, and Akash Shah. 2022. Circuit-PSI with linear complexity via relaxed batch OPPRF. *Proceedings on Privacy Enhancing Technologies* (2022).

[10] Raphaël Charbey and Christophe Prieur. 2019. Stars, holes, or paths across your Facebook friends: A graphlet-based characterization of many networks. *Network Science* 7 (2019), 476 – 497.

[11] Xiaofeng Ding, Shujun Sheng, Huajian Zhou, Xiaodong Zhang, Zhifeng Bao, Pan Zhou, and Hai Jin. 2021. Differentially Private Triangle Counting in Large Graphs. *IEEE Transactions on Knowledge and Data Engineering* 34 (2021), 5278–5292.

[12] Talya Eden, Quanquan C. Liu, Sofya Raskhodnikova, and Adam D. Smith. 2023. Triangle Counting with Local Edge Differential Privacy. *ArXiv* abs/2305.02263 (2023).

[13] Martin G Everett and Stephen P Borgatti. 1999. The centrality of groups and classes. *The Journal of mathematical sociology* 23, 3 (1999), 181–201.

[14] Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. 2021. Oblivious key-value stores and amplification for private set intersection. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part II 41*. Springer, 395–425.

[15] Chang Ge, Ihab F Ilyas, and Florian Kerschbaum. 2019. Secure multi-party functional dependency discovery. *Proceedings of the VLDB Endowment* 13, 2 (2019), 184–196.

[16] Oded Green, Pavan Yalamanchili, and Lluis-Miquel Munguia. 2014. Fast triangle counting on the GPU. In *Proceedings of the 4th Workshop on Irregular Applications: Architectures and Algorithms*. 1–8.

[17] Yunguo Guan, Rongxing Lu, Songnian Zhang, and Sean Lalla. 2023. Efficient and Privacy-Preserving Subgraph Matching Queries in Graph Federation. *ICC 2023 - IEEE International Conference on Communications* (2023), 2282–2287.

[18] Daniel Gunther, Marco Holz, Benjamin Judkewitz, Helen Mollering, Benny Pinkas, T. Schneider, and Ajith Suresh. 2022. Poster: Privacy-Preserving Epidemiological Modeling on Mobile Graphs. *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (2022).

[19] Stephen J Hardiman and Liran Katzir. 2013. Estimating clustering coefficients and size of social networks via random walk. In *Proceedings of the 22nd international conference on World Wide Web*. 539–550.

[20] Marco Holz, Benjamin Judkewitz, Helen Möllering, Benny Pinkas, and T. Schneider. 2020. PEM: Privacy-preserving Epidemiological Modeling. *IACR Cryptol. ePrint Arch.* 2020 (2020), 1546.

[21] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. 2020. Locally Differentially Private Analysis of Graph Statistics. In *USENIX Security Symposium*.

[22] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. 2022. Communication-Efficient triangle counting under local differential privacy. In *31st USENIX security symposium (USENIX Security 22)*. 537–554.

[23] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. 2022. Differentially Private Triangle and 4-Cycle Counting in the Shuffle Model. *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (2022).

[24] James D. Isaak and Mina J. Hanna. 2018. User Data Privacy: Facebook, Cambridge Analytica, and Privacy Protection. *Computer* 51 (2018), 56–59.

[25] Carter Jernigan and Behram F. T. Mistree. 2009. Gaydar: Facebook Friendships Expose Sexual Orientation. *First Monday* 14 (2009).

[26] Yuli Jiang, Xin Huang, and Hong Cheng. 2021. I/O efficient k-truss community search in massive graphs. *The VLDB Journal* 30, 5 (2021), 713–738.

[27] Leskovec Jure. 2014. SNAP Datasets: Stanford large network dataset collection. *Retrieved December 2021 from http://snap. stanford. edu/data* (2014).

[28] Vishesh Karwa, Sofya Raskhodnikova, Adam D. Smith, and Grigory Yaroslavtsev. 2011. Private analysis of graph structure. *Proceedings of the VLDB Endowment* 4 (2011), 1146 – 1157.

[29] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. 2013. Analyzing Graphs with Node Differential Privacy. In *Theory of Cryptography Conference*.

[30] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne M. Vanbriesen, and Natalie S. Glance. 2007. Cost-effective outbreak detection in networks. In *Knowledge Discovery and Data Mining*.

[31] Xue Li, Weibin Zeng, Zhibin Wang, Diwen Zhu, Jingbo Xu, Wenyuan Yu, and Jingren Zhou. 2023. GraphAr: An Efficient Storage Scheme for Graph Data in Data Lakes. *arXiv preprint arXiv:2312.09577* (2023).

[32] Yusheng Li, Yilun Shang, and Yiting Yang. 2017. Clustering coefficients of large networks. *Information Sciences* 382 (2017), 350–358.

[33] Ling Liang, Jilan Lin, Zheng Qu, Ishtiyaque Ahmad, Fengbin Tu, Trinabh Gupta, Yufei Ding, and Yuan Xie. 2023. Spg: Structure-private graph database via squeezepir. *Proceedings of the VLDB Endowment* 16, 7 (2023).

[34] Kunlong Liu and Trinabh Gupta. 2024. Making Privacy-preserving Federated Graph Analytics Practical (for Certain Queries). *Proceedings of the 29th ACM Symposium on Access Control Models and Technologies* (2024).

[35] Shang Liu, Yang Cao, Takao Murakami, Jinfei Liu, and Masatoshi Yoshikawa. 2024. CARGO: Crypto-Assisted Differentially Private Triangle Counting without Trusted Servers. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 1671–1684.

[36] Shang Liu, Yang Cao, Takao Murakami, Weiran Liu, Seng Pei Liew, Tsubasa Takahashi, Jinfei Liu, and Masatoshi Yoshikawa. 2024. Federated graph analytics with differential privacy. *arXiv preprint arXiv:2405.20576* (2024).

[37] Yuhan Liu, Suyun Zhao, Yi xiao Liu, Danting Zhao, Hong Chen, and Cuiping Li. 2022. Collecting Triangle Counts with Edge Relationship Local Differential Privacy. *2022 IEEE 38th International Conference on Data Engineering (ICDE)* (2022), 2008–2020.

[38] Mark EJ Newman. 2009. Random graphs with clustering. *Physical review letters* 103, 5 (2009), 058701.

[39] Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *Symposium on the Theory of Computing*.

[40] Xin Pang and Lanxiang Chen. 2025. Efficient and privacy-preserving butterfly counting on encrypted bipartite graphs. *Journal of Information Security and Applications* 89 (2025), 103952. https://doi.org/10.1016/j.jisa.2024.103952

[41] Tahereh Pourhabibi, Kok-Leong Ong, Booi Hon Kam, and Yee Ling Boo. 2020. Fraud detection: A systematic literature review of graph-based anomaly detection approaches. *Decis. Support Syst.* 133 (2020), 113303.

[42] Srinivasan Raghuraman and Peter Rindal. 2022. Blazing fast PSI from improved OKVS and subfield VOLE. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 2505–2517.

[43] Sofya Raskhodnikova and Adam Smith. 2016. Differentially private analysis of graphs. *Encyclopedia of Algorithms* (2016).

[44] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*. https://networkrepository.com

[45] Thomas Schank and Dorothea Wagner. 2005. Approximating clustering coefficient and transitivity. *Journal of Graph Algorithms and Applications* 9, 2 (2005), 265–275.

[46] Aakash Shah. 2021. *2PC-Circuit-PSI*. GitHub. Retrieved April 24, 2024 from https://github.com/shahakash28/2PC-Circuit-PSI

[47] Sagar Sharma, James Powers, and Keke Chen. 2019. PrivateGraph: Privacy-Preserving Spectral Analysis of Encrypted Graphs in the Cloud. *IEEE Transactions on Knowledge and Data Engineering* 31 (2019), 981–995.

[48] Julian Shun, Guy E Blelloch, Jeremy T Fineman, Phillip B Gibbons, Aapo Kyrola, Harsha Vardhan Simhadri, and Kanat Tangwongsan. 2012. Brief announcement: the problem based benchmark suite. In *Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures*. 68–70.

[49] Haipei Sun, Xiaokui Xiao, Issa M. Khalil, Yin David Yang, Zhan Qin, Wendy Hui Wang, and Ting Yu. 2019. Analyzing Subgraph Statistics from Extended Local Views with Decentralized Differential Privacy. *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (2019).

[50] Wen Jun Tan, Allan NengSheng Zhang, and Wentong Cai. 2019. A graph-based model to measure structural redundancy for supply chain resilience. *International Journal of Production Research* 57 (2019), 6385 – 6404.

[51] GraphDP Team. 2024. *CARGO*. GitHub. Retrieved April 24, 2024 from https://github.com/GraphDP/CARGO

[52] Charalampos Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria Tsiarli. 2013. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 104–112.

[53] Jia Wang and James Cheng. 2012. Truss decomposition in massive networks. *Proceedings of the VLDB Endowment* 5, 9 (2012), 812–823.

[54] Songlei Wang, Yifeng Zheng, Xiaohua Jia, Qian Wang, and Cong Wang. 2023. MAGO: Maliciously secure subgraph counting on decentralized social graphs. *IEEE Transactions on Information Forensics and Security* 18 (2023), 2929–2944.

[55] Songlei Wang, Yifeng Zheng, Xiaohua Jia, and Xun Yi. 2022. Privacy-Preserving Analytics on Decentralized Social Graphs: The Case of Eigendecomposition. *IEEE Transactions on Knowledge and Data Engineering* 35 (2022), 7341–7356.

[56] Zhibin Wang, Longbin Lai, Yixue Liu, Bing Shui, Chen Tian, and Sheng Zhong. 2023. I/o-efficient butterfly counting at scale. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–27.

[57] Zhibin Wang, Longbin Lai, Yixue Liu, Bing Shui, Chen Tian, and Sheng Zhong. 2024. Parallelization of butterfly counting on hierarchical memory. *The VLDB Journal* 33, 5 (2024), 1453–1484.

[58] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world'networks. *nature* 393, 6684 (1998), 440–442.

[59] Mark Weber, Jie Chen, Toyotaro Suzumura, Aldo Pareja, Tengfei Ma, Hiroki Kanezashi, Tim Kaler, Charles E. Leiserson, and Tao B. Schardl. 2018. Scalable Graph Learning for Anti-Money Laundering: A First Look. *ArXiv* abs/1812.00076 (2018).

[60] Ruidi Wei and Florian Kerschbaum. 2023. Cryptographically secure private record linkage using locality-sensitive hashing. *Proceedings of the VLDB Endowment* 17, 2 (2023), 79–91.

[61] Minze Xu, Yuan Zhang, Fengyuan Xu, and Sheng Zhong. 2021. Privacy-preserving optimal recovering for the nearly exhausted payment channels. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*. IEEE, 1–10.

[62] Minze Xu, Yuan Zhang, and Sheng Zhong. 2024. Towards Payment Channel Watchtowers with Collateral-free Security and Robustness. *IEEE Transactions on Dependable and Secure Computing* (2024).

[63] Jaewon Yang, Julian McAuley, and Jure Leskovec. 2013. Community detection in networks with node attributes. In *2013 IEEE 13th international conference on data mining*. IEEE, 1151–1156.

[64] Feng Yao, Qian Tao, Wenyuan Yu, Yanfeng Zhang, Shufeng Gong, Qiange Wang, Ge Yu, and Jingren Zhou. 2023. Ragraph: A region-aware framework for geo-distributed graph processing. *Proceedings of the VLDB Endowment* 17, 3 (2023), 264–277.

[65] Yuchen. 2021. *Kunlun*. GitHub. Retrieved April 24, 2024 from https://github.com/yuchen1024/Kunlun

[66] Ke Zhong and Sebastian Angel. 2024. Oryx: Private detection of cycles in federated graphs. *Cryptology ePrint Archive, Paper 2024/1117* (2024).

[67] Zeqi Zhu, Zeheng Fan, Yuxiang Zeng, Yexuan Shi, Yi Xu, Mengmeng Zhou, and Jin Dong. 2024. FedSQ: A Secure System for Federated Vector Similarity Queries. *Proceedings of the VLDB Endowment* 17, 12 (2024), 4441–4444.