LiPI: Lightweight Privacy-Preserving Data Aggregation in IoT

Himanshu Goyal, Krishna Kodali, Sudipta Saha
School of Electrical Sciences, Indian Institute of Technology Bhubaneswar, India
Email: {hg11, knk11, sudipta}@iitbbs.ac.in

Abstract—The ability of IoT devices to sense and share various physical parameters plays a key role in a smart system. Alongside benefits, it also bears the potential to cause a breach of privacy for the users of the smart-systems. Existing solutions for privacy-preserving computation for distributed systems either extensively use highly complex cryptographic techniques or exploit an extremely high degree of message passing among the devices through secure channels. However, for the resource-constrained IoT devices, which compose a significant fraction of the smart-systems, the existing solutions for privacy-preserving computation strategies do not fulfill the current requirements. To address this issue, in this work, we propose a novel real-time lightweight strategy LiPI for Privacy-Preserving Data Aggregation in low-power IoT systems. LiPI uses lightweight distributed and collaborative data obfuscation, which substantially minimizes the computation requirements. In addition, it exploits the recent advances in Synchronous-Transmission (ST)-based protocols to efficiently fulfill the communication requirements too, making it efficient to work in real-time. Furthermore, LiPI also avoids dependency on any trusted third party. Extensive evaluation based on comprehensive experiments in both simulation platforms and publicly available WSN/IoT testbeds demonstrates that our strategy achieves the goal at least 51.7% faster and consumes 50.5% lesser energy compared to the existing state-of-the-art strategies.

Index Terms—Privacy-Preserving Data Aggregation, Collaborative Obfuscation, Synchronous Communication, Concurrent-Transmissions, Internet of Things (IoT), Wireless Sensor Networks.

I. INTRODUCTION

In today’s modern digital world, many vital and significant aspects of human lives are being substantially driven by IoT/WSN-assisted smart systems, e.g., Smart-Health-Care [1], Wireless-Body-Area Networks [2], Smart-Grid [3], Multi-Robot Collaboration [4] and many others. The data sensed by the devices used in such smart-systems often bear a direct relationship with the private/sensitive information of the users of the system. For instance, in Advanced Metering Infrastructure (AMI) [5], the electricity consumption data collected from a house can be used to precisely infer the activities inside the house [6]. Leakage of the values of the raw physical parameters sensed by the IoT devices used in the deployment of the Wireless Body Area Network (WBAN) can also be used to infer various vital health-related status of the corresponding patient. Similarly, the data shared by a set of robots while executing a collaborative task make the system always vulnerable to targeted attacks.

Data aggregation is a very simple, common, and frequently executed operation in any smart system. However, any such aggregation operation needs sharing of the data sensed by an IoT device with others. To avoid breach of privacy, IoT applications avoid direct use of the raw data. A Privacy-Preserving Data-Aggregation (PPDA) strategy used in an IoT system adopts various privacy-preserving sound strategies to appropriately transform the data before a device shares it with others, and, later, uses the transformed data to carry out the aggregation task. Naturally PPDA strategies become quite complicated. They need to employ quite a few additional steps to maintain the secrecy of the data and carry out the necessary aggregation task over the transformed data without compromising the correctness. However, a significant fraction of the devices used in an IoT system is resource-constrained. The use of complex PPDA strategies in such a system may either be infeasible to execute or results in a system with extremely slow behavior. Thus, although PPDA has been a well-studied topic for traditional systems, its use in low-power IoT systems is still challenging.

A major fraction of the existing works on PPDA are based on Homomorphic Encryption (HE) [7] which is highly computation intensive, and hence, is not that suitable for resource-constrained IoT-edge devices. Another class of works are based on Multi-Party Computation (MPC) [8] which are highly communication intensive, and hence, do not directly fit with the low-power IoT-edge devices. Data Obfuscation (DO) has been also another tool for accomplishing PPDA [9], [10], [11]. It involves less computation and, hence, fit well with the IoT-edge smart systems. However, generation of noise to achieve data-hiding in DO is usually accomplished by the help of Trusted Third Party (TTP) which makes the whole process quite vulnerable to breach of privacy, and also results in higher end-to-end latency.

In this work, we propose LiPI, a novel lightweight solution for PPDA in low-power IoT systems. It is fundamentally based on simple DO with minimized use of cryptography. LiPI is designed as a self-sufficient strategy that does not depend on any TTP or any centralized entity for any purpose linked with the key generation/noise generation. Instead, LiPI makes extensive use of distributed computation and appropriate collaboration among the edge-devices to accomplish its goals. Furthermore, to minimize the communication overhead, LiPI is designed to exploit the recent advancements in Synchronous-Transmission (ST) to efficiently execute tasks in resource-constrained IoT devices in a highly time and energy-efficient manner.

The main contributions of the proposed work are summarized below.

• We design a simple lightweight strategy LiPI for Pri-
We propose an ST-based framework to efficiently share different parts with different nodes.

- We implement LiPI and compare it with state-of-the-art strategies under ST-based frameworks in Contiki OS for TelosB devices and rigorously evaluate their performance in both simulation and publicly available WSN/IoT Testbeds.

The rest of the paper is organized as follows. Section III provides a brief description of the ST-based protocols that are used to construct LiPI. Section IV provides the detailed design of LiPI. Section V reports the evaluation of LiPI in comparison with the other existing strategies.

II. RELATED WORKS

Homomorphic Encryption (HE) [7] has been one of the primary tools to accomplish PPDA. It enables computation of aggregation operations directly over cipher-text alleviating the requirement of the intermediate deciphering of the data. However, in many HE-based works [12], [13], [14], the sink (i.e., the final destination of the data) is assumed to be trustworthy and is provided with the key to decipher the final result, which also enables it to decipher the individual cipher texts from the individual nodes. To resolve this, the work (PPEPPDA) [13] uses a tree structure and dynamic slicing of the data. In contrast, the work (3PDA) [14] uses an intermediate Data Collection Unit to achieve the same. However, apart from these known issues, in general, the execution of HE in every step of aggregation process is quite computationally intensive, making it unsuitable for low-power IoT systems.

Multi-Party Computation (MPC) has also been adopted for solving PPDA by a set of works. The concept of Secure Multi-Party Computation (SMPC) was introduced by the work [8]. Shamir’s Secret Sharing [15] is one of the most adopted strategies to achieve SMPC in decentralized systems. In SMPC, each source node first divides its secret value into multiple parts and shares different parts with different nodes. Finally, the node runs the second round of data-sharing to derive the final result. The algorithms CPDA and SMART proposed in work [16] improve upon these basic concepts through the application of secret-splitting and collaborative computing. Thus, MPC-based approaches exploit less computation but incur heavy communication over secure channels, which is also a serious concern in the context of energy-constrained IoT systems.

Furthermore, appropriate obfuscation of the data (referred to as Data Obfuscation or DO) with the help of random noise has also been a very standard way of hiding the original data to achieve PPDA [9], [10], [11]. DO is considered a computationally much less intensive operation than using cryptography to transform the data. However, the noise is supposed to be generated in a very organized way so that it can be effectively removed to obtain the correct aggregation. Most of the existing DO-based works depend on some Trusted Third Party (TTP) for sharing the necessary random noise to the nodes’ values, making them unreliable. Some of the works, e.g., DPPDA [11], generate the noise values without the help of any TTP and achieve accurate convergence. However, their applicability is quite limited, e.g., DPPDA works only for finding averages. Moreover, these solutions also exploit the knowledge of the network topology, which may not be available in a generic setting.

In this work we introduce a novel way to accomplish PPDA in resource constraint IoT-edge. We approach with DO where the random noise is computed in a collaborative way so that the strategy works in distributed and independent fashion.

III. BACKGROUND

Sharing of data among the nodes is the key component in any collaborative/distributed algorithm. Existing DO or MPC-based works mostly use traditional Asynchronous-Transmission (AT) based strategies to serve the purpose. However, due to mostly uncoordinated and unplanned transmissions, with even a little rise in the data traffic, the chance of packet collision rapidly shoots up, causing drastic degradation in the performance of the protocol under AT. Such issues make the use of AT highly inappropriate where the nodes need to have a quite high degree of interaction in real-time.

In many recent works, ST-based strategies have been shown to be superior to AT-based ones in terms of reliability, latency, and energy consumption [17]. In contrast to AT, the protocols under ST try to coordinate the transmissions from the participating nodes in such a way that the packets originating from different source nodes, instead of colliding with each other, successfully exploit certain physical layer phenomena, e.g., Capture-Effect (CE)/Constructive-Interference (CI) and get correctly received at the destination nodes. Such measures enable the ST-based strategies to save a lot of time and energy because of less use of RF units in the devices. ST has been used in various all-to-all/many-to-many data-sharing protocols in IoT/WSN [18], [19], [20], [21]. In this work, as a base of the proposed strategy LiPI, we use a protocol MiniCast [21] which is a quite simple and efficient alternative for many-to-many data sharing. For completeness, below, we provide a very brief description of the fundamental ST-based mechanisms.

Glossy: Time-synchronization is the first and foremost requirement for any ST-based strategy. The work Glossy [22] shows how the same can be achieved through purely in-network RF-based communication. Besides precise time-synchronization, Glossy also shows how to achieve efficient network-wide flooding, i.e., one-to-many data dissemination in resource-constrained IoT/WSN systems. Glossy is a prerequisite before running MiniCast, which is described next.

MiniCast: MiniCast fundamentally extends the functionality of Glossy by enabling sharing of data from multiple or all the source nodes in an efficient and highly compact form through the use of TDMA. In particular, while Glossy allows a single node to share its data with the other nodes, MiniCast uses a schedule at a very fine-grained level so that multiple flooding operations happen almost at the same time in parallel, like a pipelined execution of multiple Glossy floods. Thus, while the unit of transmission in Glossy is a single packet, in MiniCast its is a chain of packets where each one
is contributed by different node. Time for transmission or reception of a chain of packets is referred to as a slot. The unique position allocated to each node in a slot is called a sub-slot.

IV. DESIGN

The primary aim of the work is to device a simple strategy that can correctly carry out aggregation of the data-items shared by the IoT-devices in a way so that devices can not see each others data (privacy). We assume a semi-honest adversarial model where every node in the system is supposed to follow the protocol specification correctly. However, the honest-but-curious or passive adversaries are free to learn the information from the internal states of the other nodes.

In a nutshell, we use a collaborative DO, where a node, before sharing the data for global aggregation, changes the same by augmenting with some noise which is referred to as masking. However, the noise data is calculated through a joint discussion among the nodes so that when a node combines all the data from all the nodes, the augmented noise data cancels each other which is referred to as demasking. In the following we first provide the details of the masking and demasking procedure. Subsequently we describe the design of the complete system.

A. Masking and De-masking

Algo. 1 provides a design sketch of LiPI. Let \( S_i \) be the private data of node \( N_i \). The target is to compute a function \( f(S_1, S_2, ..., S_n) = S \) in each node without allowing a node to know the private data of any other node. Every node first obfuscates \( S_i \) using a function \( f_1 \) (masking function) and obtains \( M_i \). In other words, \( M_i = f_1(S_i, q_{i1}, ..., q_{in}) \), where \( q_{ij} \)'s are the noise values which are added by node \( N_i \) for every other source node \( N_j \) in the system. In the subsequent MiniCast round, each node shares \( M_i \) and on completion, receives a vector \( < M_1, M_2, ..., M_i, ..., M_n > \). Finally, the function \( f_2 \) (de-masking function) is applied over this vector to produce the final outcome, i.e., the target joint aggregation function \( f \). Conversely, \( S = f_2(M_1, M_2, ..., M_i, ..., M_n) = f(S_1, S_2, ..., S_i, ..., S_n) \). This procedure is shown in the Part-2 (Aggregation) of Algo. 1.

Collaborative-obfuscation: Every node independently computes a noise value for every other node. \( f_1 \) generates a single value \( M_i \) combining the noise and the own private data at node \( N_i \), which is shared through MiniCast with other nodes. The noise quantities are computed in such a way that, when all these \( M_i \)'s are collected together, the application of \( f_2 \) locally in each node can result in their mutual cancellation and hence, a correct computation of the aggregation. For example, if node \( N_i \) contributes noise \( q_{ij} \) for node \( N_j \), and node \( N_j \) contributes \( q_{ji} \) for node \( N_i \), during the computation of \( f_2 \), \( q_{ji} \) is supposed to cancel \( q_{ij} \).

Generating noise values: For an automated and independent generation of noise values without using any TTP, LiPI makes use of pre-decided pair-wise secret keys. Deciding pair-wise secret keys is a computationally expensive job. However, LiPI exploiting the underlying ST-based framework involves this step as less as possible (very rarely) and mostly on-demand basis. In particular, during a bootstrapping phase, pair-wise secret keys \( P_{ij} \) are settled through the procedure explained in Algo. 2. Next, for all the next iterations of computation of the aggregation, LiPI first uses a system-wide agreed Pseudo Random Number Generator (PRNG) function seeded by MiniCast iteration number (i.e., seq_no) concatenated with \( P_{ij} \) to generate a value \( r_{ij} \), i.e., \( r_{ij} = PRNG(P_{ij}||seq_no) \). Finally, to ensure appropriate cancellation of the noise values during the application of \( f_2 \), a third function \( f_3 \), is used to derive the final noise values \( q_{ij} \) from \( r_{ij} \), i.e., \( q_{ij} = f_3(r_{ij}) \). Part-1 (Pre-processing) of Algo. 1 describes this procedure. The exact form of \( f_3 \) depends on both \( f_1 \) and \( f_2 \). Use of an ST-based framework ensures that at any iteration, MiniCast seq_no is the same in all the nodes in the system.

B. Pair-wise secret-key

An adversary can reveal private data shared by a node if and only if it can infer the individual noise values. The use of pairwise secret keys is to make this hard or impossible. To avoid the use of any TTP, LiPI generates the pairwise keys in a collaborative way by exploiting the well-known Diffie-Hellman Key Exchange (DFKE) [23]. However, to settle all the \( N \) pairs of keys, LiPI exploits the ST-based framework. Algo. 2 narrates the main key-exchange process. In a nutshell, the parameter \( q \) and a prime number \( p \) are globally decided. Moreover, each node \( N_i \) locally decides a secret value \( d \) and computes \( V_i = q^d \ mod \ p \), which are next shared among each other through a single round of MiniCast. Finally, these values are used as per the standard DFKE process to compute the N-1 pair-wise secret keys in each node.

Once a set of keys are generated, they are reused through PRNG, which is quite a lightweight task, and now a days most of the SoC (System-on-Chip) offers specialised hardware for such operations. Key generation is invoked only when a new node joins or after a certain number of usage of the previously generated keys. The use of PRNG guarantees that an adversary not knowing the seed has only a negligible advantage in distinguishing the generator’s output sequence from a random sequence.

V. EVALUATION

Experimental-Setup: We implement LiPI in the Contiki Operating System for TelosB devices and extensively experiment in both Contiki Network Simulator Cooja [24] as well as publicly available IoT/WSN testbeds FlockLab [25] and DCube[26], which contains 24 and 47 TelosB devices, respectively. DCube testbed has two loosely connected islands containing 31 and 16 nodes. To isolate the possible reliability issues related to the execution of network protocols in all our experiments, we mostly use the larger island, having 31 nodes.

A. Parameters and Metric

The parameter NTX in MiniCast determines how many times a node forwards or transmits the chain of packets. In ST-based protocols, NTX determines mainly the reliability. However, in MiniCast, since it starts the floods from different source nodes, a less reliable operation implies a limited outreach of each individual flood. The work NSSS [27] exploits
Output:

$q \in \text{Prepare}$

/*
IN SET OF NODES
← NLINE
← \forall N
/*
IN SET OF NODES
$q \leftarrow \text{Execute all-to-all data-sharing using MiniCast.}
\forall \ Output:
0 \in = N
N
Input:
P
for N
Communication protocol ends

\forall \ Output:
\leftarrow \text{Obtains a vector}
P
N
r
N
r
S
1t oN
no
/*
Initialise:
L
Initiate a Glossy flood to share
and
Decide a secret value
N
g
Input
)
N
is computed as
and
/*
Decide a prime number
g
Latency:
\begin{align*}
E & = \text{Obtains a vector} \\
& = \text{P}
\end{align*}
Communication protocol ends
locally computes the value
P
Output = ( \) and place in MiniCast chain

Algorithm 1 ALGORITHMIC DESCRIPTION OF LiPI

PART-1: (PRE-PROCESSING: OFFLINE PHASE)

\textbf{EVERY NODE } N, \textbf{IN SET OF NODES } X

1. Input: \( P_i \forall N_i \in X \setminus N_i, \text{seq}_n, f_3 \)
2. Output: \( q_{ij} \forall N_i \in X \setminus N_i \)
3. Initialise: \( r_{ij} \leftarrow 0, q_{ij} \leftarrow 0 \)
4. for \( j \leftarrow 1 \) to \( N \in X \setminus N_i \) do
5. \( r_{ij} \leftarrow \text{PRNG}(P_i || \text{seq}_n) \),
6. \( q_{ij} \leftarrow f_3(r_{ij}) \)
7. end for

PART-2: (AGGREGATION: ONLINE PHASE)

\textbf{EVERY NODE } N, \textbf{IN SET OF NODES } X

1. Input: Secret value \( S_i \), Noise \( q_{ij} \forall N_j \in X \setminus N_i, f_1, \) and \( f_2 \)
2. Output: \( S = f_1(S_i, q_1, q_2, q_3, ..., q_n) \)
3. /* Local computation */
4. Calculate \( M_i = f_1(S_i, q_1, q_2, q_3, ..., q_n) \)
5. /* Communication protocol starts */
6. Prepare a packet with value \( M_i \) and place in MiniCast chain
7. Execute all-to-all data-sharing using MiniCast.
8. /* Stops after completion of NTX transmission of chain */
9. Obtains a vector \( < M_1, M_2, ..., M_i, ..., M_n > \)
10. /* Communication protocol ends */
11. /* Local computation */
12. \( S \leftarrow f_2(M_1, M_2, ..., M_n) \)

Algorithm 2 DF-KEY EXCHANGE (DFKE)

\textbf{INITIATOR/SINK}

1. Decide a prime number \( p \) and a primitive root modulo of the prime number \( g \).
2. Initiate a Glossy flow to share \( p \) and \( g \) with all other nodes.

\textbf{EVERY SOURCE NODE } N, \textbf{IN SET OF NODES } X

1. Input: Receive \( p \) and \( g \).
2. Decide a secret value \( d_i \), and compute \( V_i = g^{d_i} \mod p \).
3. Output: \( \forall N_j, \, N_j \in N \leftrightarrow P_i \)
4. /* Communication protocol starts */
5. Prepare a packet with value \( V_i \) and place in MiniCast chain
6. Execute all-to-all data-sharing using MiniCast.
7. /* Stops after completion of NTX transmission of chain */
8. Obtains a vector \( < V_1, V_2, ..., V_i, ..., V_n > \)
9. /* Communication protocol ends */
10. For every other node \( N_j \) locally computes the value \( (V_j)^{d_i} \mod p \).
11. The common secret keys \( P_{ij} \) shared between each pair of nodes, \( N_i \) and \( N_j \) is computed as \( P_{ij} = ((V_j)^{d_i}) \mod p = ((V_i)^{d_i}) \mod p = (g^{d_i \cdot d_j}) \mod p \).

The metrics used to measure the efficiency and robustness of LiPI and other strategies are described below.

- **Latency**: It is the time taken for a node to obtain the final aggregation value. It considers both the computation and communication phases of the protocol.
- **Radio-on time**: It is the total time for which a node keeps its radio ON for the data sharing operation. Radio-on time depends on the value of the NTX. In resource-constrained IoT devices, Radio-on time directly reflects the energy consumption, as the radio (RF unit) is supposed to be

![Fig. 1. Latency (Part a,c) and Radio-on time (Part b,d) in executing LiPI-based calculation of sum values in IOT/WSN setting comprised of the different number of nodes and spread over different areas in Cooja.](image)

![Fig. 2. Percentage of data received and the intermediate aggregated value after each chain reception](image)

**B. Basic study**

We first experiment with LiPI in Cooja. Each experiment is executed for at least 1000 iterations, and the metrics are computed as an average over all the iterations and all the nodes. The error bars in the results reflect the standard deviation. For simplicity in all the evaluation experiments, we assume the target aggregation function to be the summation of all the secret values. In any ST-based protocol, an initiator (here referred to as sync) node starts the whole process. In general, any node can work as an initiator, and a new initiator can get elected when the existing one fails. However, because of its special role, in all the results, we separately report the Latency and Radio-on time of the initiator/sync.

**Effect of the number of nodes**: LiPI is first executed in Cooja with 10, 30, 50, and 70 no. of nodes randomly spread over area of different size. Fig. 1(a) and 1(c) show the Latency in both sync and other nodes for two different areas 500 \( \times \) 500, and 1000 \( \times \) 1000 (all in \( m^2 \)) respectively. Radio-on-time values are shown in 1(b) and 1(d). Almost a linear increase in
the metrics with the number of participants is visible in both cases. However, with the increase in the deployment area, the diameter of the network increases, which makes the underlying data-sharing process to put more effort into executing all-to-all data-sharing. This is reflected in the considerable rise in the Latency and Radio-on time for the 70-node network in 1000x1000 sq. meters compared to the other settings.

C. Comparison of LiPI with other PPDA strategies

In order to compare the performance of LiPI, we select three different state-of-the-art PPDA strategies, namely, PPMP [28], SSS [29] and NSSS [27]. However, none of the works provide any implementation for low-power hardware devices. In addition, there is no attempt so far to incorporate ST based framework to realize any PPDA strategy. Therefore, to make a fair comparison, we implement all the three strategies in the same ST-based framework that we use for LiPI. In the following, we describe them in detail.

1) PPMP: PPMP satisfies a significant fraction of the properties that are expected from a PPDA strategy and are targeted in this work, too. However, it assumes a special circular arrangement of the nodes where every node is assigned with exactly two adjacent nodes (referred to as \((i + 1)\) and \((i - 1)\)). The whole task is carried out in two rounds of communication/data-sharing, referred to as the key exchange and the aggregation.

We implement PPMP using MiniCast. In the key-exchange round, PPMP needs to maintain a circular arrangement where every node is supposed to share the keys with its two adjacent nodes (left and right).

The key-exchange round is executed through a MiniCast-based all-to-all data-sharing. Subsequently, the aggregation is executed through another complete round of MiniCast. DFKE in LiPI can be compared with the key-exchange round in PPMP. Although both need the same communication effort, in LiPI, it is enough to execute DFKE once after several rounds of aggregation. However, in PPMP, the fresh key exchange has to precede every aggregation round which makes PPMP highly computation and communication intensive.

2) Shamir’s Secret Sharing (SSS): SSS has been used in many works for SMPC, which can be directly used for PPDA. In SSS, every node first decides a local n-degree polynomial \(P_i(x)\) where \(n\) is the number of nodes in the system. The secret key \(S_i\) of \(N_i\) is considered to be the constant terms in \(P_i\), i.e., \(P_i(0)\). \(P_i\) is first evaluated at \(n\) distinct predefined points \((y_1, y_2, y_3, \ldots, y_n)\) and the values are next shared with the nodes in the network using secure channel (i.e., \(P_i(y_j)\) is shared by \(N_i\) only with \(N_j\) through a secured channel). To maintain uniformity, we implement SSS using MiniCast, where each node puts data in \((n - 1)\)-distinct sub-slots encrypted (AES-128) using pre-agreed pair-wise secret keys (we use DFKE for settling the pairwise keys). After completion of MiniCast, every node \(N_i\) decrypts all the \(n - 1\) values received and sum up locally to derive \(K_i = \sum_{j=1}^{n} P_i(y_j))\). Next, the second round of MiniCast (reconstruction-round) is used to share the values of \(K_i\) among each other in plain text. This finally constructs a polynomial \(P\) using Lagrange-interpolation in every node which represents the sum of all the polynomials adopted by all the participating nodes. Thus the constant term in \(P\) happens to be the sum of the secret values i.e., \(\sum_{i=1}^{n} S_i\).

3) Neighborhood-based Shamir’s Secret Sharing (NSSS): Implementation of SSS, even through our proposed ST-based strategy, is extremely communication intensive. In particular, the size of the individual chain used in SSS is \(O(n^2)\) as it uses all-to-all communication for both data-sharing and reconstruction-round. The work [27] demonstrates an optimized version of SSS where a node instead of interacting with everyone in the network, only involves its neighbors. Specifically, it assumes a \(d\)-degree polynomial of degree lesser than \(n\), i.e., the total number of nodes, in such a way that in the data-sharing round, it is sufficient for a node to share its data only with its neighbors. To accomplish this, the parameter NTX in MiniCast is appropriately adjusted to tame the outreach of the propagation of the data from each node only up to a few hops. This substantially reduces the communication overhead compared to SSS. We refer to this version as Neighborhood based SSS (NSSS). However, it still requires encryption of packets before transmission in the sharing phase of MiniCast. Moreover, the re-construction phase also requires an all-to-all interaction among the participants.

Encryption: Both SSS and NSSS require explicit encryption of data in each and every step to implement a secure channel between pairs of nodes. In contrast, as already described, LiPI and PPMP do not need any encryption and solely depend on lightweight DO.

Collusion: SSS and LiPI have the highest collusion resilience (collusion-threshold is \(n-1\), and \(n-2\) respectively) as both mandate the participation of almost all the nodes in the computation of the aggregation. NSSS improves over SSS by restricting the communication only among the neighbors. Although it makes the protocol simpler, collusion threshold is decreased. PPMP exhibits the least resilience against collusion. Because of the circular arrangement of the nodes in PPMP, a node shares data with two other nodes. Let us consider every node \(N_i\) shares \(C_i = (1 + x_i * p) \ast (g^{r_{1}})^{x_{i}}/g^{(r_{1} + 1)}\) (where \(x(i)\) is the private data) with its two adjacent nodes. Now, the adjacent nodes can collude with each other to calculate \((g^{r_{1}})^{-1} / (g^{r_{1} + 1})\) as \(r_{1} + 1\) and \(r_{1} - 1\) are private to them and \(g^{r_{1}}\) is already received in the key-exchange round. Thus, in PPMP, the collusion threshold is just 1.

Data-sharing complexity: Finally, LiPI substantially minimizes the communication requirement. Ideally, it just needs a single round of MiniCast-based all-to-all data-sharing. However, in all the other strategies, even when there is no node failure, it requires multiple rounds of MiniCast. Moreover, due to the use of restricted data-sharing, a node needs to reserve multiple sub-slots in MiniCast, which substantially increases the cost of communication.

Validation in Testbed: We validate the simulation and theoretical findings through extensive studies with two IoT/WSN testbeds. All the four PPDA strategies are executed to obtain the sum of the randomly assumed private values by the nodes. A comparison of the performance of the protocols in terms of Latency and Radio-on time is depicted in Fig. 3. It can
be observed that LiPI outperforms all the other strategies in both metrics. In FlockLab, on average, LiPI achieves PPDA up to 51.7%, 68.56%, and 81.91% faster and consumes 50.49%, 66.49%, and 80.35% lesser energy (lesser use of RF-unit) compared to PPMP, NSSS, and SS, respectively. Similarly, in DCube, it accomplishes the task up to 52.04%, 71.41%, and 87.3% faster and consumes 51.72%, 70.89%, and 86.7% lesser energy compared to PPMP, NSSS, and SS, respectively.

VI. CONCLUSION

Data aggregation is a very frequently used operation in any distributed system in general. However, carrying out aggregation, either with the help of a centralized entity or in a distributed fashion, need the nodes to share their data, which naturally incurs the possibility of a breach of privacy. Existing solutions for privacy-preserving data aggregation mostly involve computation-intensive homomorphic encryption, complex data obfuscation, or even communication-intensive multi-party computation. These solutions are, hence, too heavy to be useful or even applicable for the low-power devices which compose a significant part of an IoT-assisted smart system. In this work, we propose a simple, lightweight strategy LiPI for privacy-preserving-data-aggregation through collaborative data obfuscation. We also propose a Synchronous-Transmission based framework where we implement LiPI and three other existing state-of-the-art strategies. Through extensive simulation and testbed-based experiments, we demonstrate that LiPI outperforms the existing state-of-the-art strategies in several aspects by a large margin.

ACKNOWLEDGMENT

The authors would like to thank Indian Institute of Technology Bhubaneswar for the financial support. They are also thankful to the TU Graz, and ETH Zurich community for offering the public access to the physical testbeds for the experimentation purposes. Any opinions, findings and conclusions or recommendations contained in this material are those of the author(s) and do not reflect the views of IIT Bhubaneswar, India.

REFERENCES

[1] S. Tyagi, A. Agarwal, and P. Maheshwari, “A conceptual framework for iot-based healthcare system using cloud computing,” in IEEE Conference, 2016.
[2] T. Wu, F. Wu, J.-M. Redouté, and M. R. Yuce, “An autonomous wireless body area network implementation towards iot connected healthcare applications,” IEEE Access, 2017.
[3] Y. Saleem, N. Crespi, M. H. Rehmani, and R. Copelan, “Internet of things-aimed smart grid: Technologies, architectures, applications, prototypes, and future research directions,” IEEE Access, 2019.
[4] P. Mahato, S. Saha, C. Sarkar, and M. Shaghil, “Consensus-based fast and energy-efficient multi-robot task allocation,” Robotics and Autonomous Systems, 2023.
[5] H. Mohammed, S. Tonyali, K. Rabieh, M. Mahmoud, and K. Akkaya, “Efficient privacy-preserving data collection scheme for smart grid ami networks,” in IEEE GLOBECOM, 2016.
[6] S. Desai, R. Alhaidal, N. Chalamburti, and A. Mahmood, “A survey of privacy preserving schemes in ioe enabled smart grid advanced metering infrastructure,” Cluster Computing, 2019.
[7] C. Gentry, A fully homomorphic encryption scheme, 2009.
[8] A. C. Yao, “Protocols for secure computations,” in 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982), 1982.
[9] W. Jia, H. Zhu, Z. Cao, X. Dong, and C. Xiao, “Human-favor-aware privacy-preserving aggregation in smart grid,” IEEE Systems Journal, 2014.
[10] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, “Our data, ourselves: Privacy via distributed noise generation,” in Advances in Cryptology - EUROCRYPT 2006. Springer Berlin Heidelberg, 2006.
[11] J. He, L. Cai, P. Cheng, J. Pan, and L. Shi, “Distributed privacy-preserving data aggregation against dishonest nodes in network systems,” IEEE Internet of Things Journal, 2019.
[12] J. Zhang, Y. Zhao, J. Wu, and B. Chen, “Lypda: A lightweight and verifiable privacy-preserving data aggregation scheme for edge-enabled iot,” IEEE Internet of Things Journal, 2020.
[13] J. Jose, M. Princy, and J. Jose, “Peppda: Power efficient privacy preserving data aggregation for wireless sensor networks,” in IEEE ICECCN, 2013.
[14] Y. Liu, W. Guo, C.-I. Fan, L. Chang, and C. Cheng, “A practical privacy-preserving data aggregation (3pda) scheme for smart grid,” IEEE Transactions on Industrial Informatics, 2019.
[15] A. Shamir, “How to share a secret,” Commun. ACM, 1979.
[16] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher, “Pda: Privacy-preserving data aggregation in wireless sensor networks,” in IEEE INFOCOM, 2007.
[17] M. Zimmerling, L. Mottola, and S. Santini, “Synchronous transmissions in low-power wireless: A survey of communication protocols and network services,” ACM Computing Surveys (CSUR), 2020.
[18] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, “Low-power wireless bus,” in SenSys, 2012.
[19] O. Landsiedel, F. Ferrar, and M. Zimmerling, “Chaos: Versatile and efficient all-to-all data sharing and in-network processing at scale,” in SenSys, 2013.
[20] M. Mohammad and M. C. Chan, “Codecast: Supporting data driven in-network processing for low-power wireless sensor networks,” in IPSN, 2018.
[21] S. Saha, O. Landsiedel, and M. C. Chan, “Efficient many-to-many data sharing using synchronous transmission and timed,” in DCOS, 2017.
[22] F. Ferrari, M. Zimmerling, L. Thiele, and O. Sankh, “Efficient network flooding and time synchronization with glossy,” in IPSN, 2011.
[23] W. Diffie and M. E. Hellman, “New directions in cryptography,” in Secure communications and asymmetric cryptosystems, 1976.
[24] S. Tyagi, A. Agarwal, and P. Maheshwari, “A conceptual framework for iot-based healthcare system using cloud computing,” in IEEE Conference, 2016.
[25] A. C. Yao, “Protocols for secure computations,” in 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982), 1982.
[26] W. Jia, H. Zhu, Z. Cao, X. Dong, and C. Xiao, “Human-favor-aware privacy-preserving aggregation in smart grid,” IEEE Systems Journal, 2014.
[27] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, “Our data, ourselves: Privacy via distributed noise generation,” in Advances in Cryptology - EUROCRYPT 2006. Springer Berlin Heidelberg, 2006.
[28] J. He, L. Cai, P. Cheng, J. Pan, and L. Shi, “Distributed privacy-preserving data aggregation against dishonest nodes in network systems,” IEEE Internet of Things Journal, 2019.
[29] J. Zhang, Y. Zhao, J. Wu, and B. Chen, “Lypda: A lightweight and verifiable privacy-preserving data aggregation scheme for edge-enabled iot,” IEEE Internet of Things Journal, 2020.
[30] J. Jose, M. Princy, and J. Jose, “Peppda: Power efficient privacy-preserving data aggregation for wireless sensor networks,” in IEEE ICECCN, 2013.
[31] Y. Liu, W. Guo, C.-I. Fan, L. Chang, and C. Cheng, “A practical privacy-preserving data aggregation (3pda) scheme for smart grid,” IEEE Transactions on Industrial Informatics, 2019.
[32] A. Shamir, “How to share a secret,” Commun. ACM, 1979.
[33] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher, “Pda: Privacy-preserving data aggregation in wireless sensor networks,” in IEEE INFOCOM, 2007.
[34] M. Zimmerling, L. Mottola, and S. Santini, “Synchronous transmissions in low-power wireless: A survey of communication protocols and network services,” ACM Computing Surveys (CSUR), 2020.
[35] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, “Low-power wireless bus,” in SenSys, 2012.
[36] O. Landsiedel, F. Ferrar, and M. Zimmerling, “Chaos: Versatile and efficient all-to-all data sharing and in-network processing at scale,” in SenSys, 2013.
[37] M. Mohammad and M. C. Chan, “Codecast: Supporting data driven in-network processing for low-power wireless sensor networks,” in IPSN, 2018.
[38] S. Saha, O. Landsiedel, and M. C. Chan, “Efficient many-to-many data sharing using synchronous transmission and timed,” in DCOS, 2017.
[39] F. Ferrari, M. Zimmerling, L. Thiele, and O. Sankh, “Efficient network flooding and time synchronization with glossy,” in IPSN, 2011.
[40] W. Diffie and M. E. Hellman, “New directions in cryptography,” in Secure communications and asymmetric cryptosystems, 1976.
[41] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, “Cross-level sensor network simulation with cooja,” in IEEE LCN, 2006.
[42] R. Trüb, R. Da Forno, L. Sigrist, L. Mühlebach, A. Biri, J. Beutel, and L. Thiele, “FlockLab 2: Multi-Modal Testing and Validation for Wireless IoT,” in CPS-IoTBench 2020.
[43] M. Schüß, C. A. Bouno, M. Weber, and K. Römer, “A competition to push the dependability of low-power wireless protocols to the edge,” in IEEE EWSN, 2017.
[44] H. Goyal and S. Saha, “Multi-party computation in iot for privacy-preservation,” in IEEE ICDCS, 2022.
[45] T. Jung, X. Mao, X.-Y. Li, S.-J. Tang, W. Gong, and L. Zhang, “Privacy-preserving data aggregation without secure channel: Multivariate polynomial evaluation,” in 2013 Proceedings IEE INFOCOM, 2013.
[46] G. S. Wagh and S. Mishra, “Divide & conquer: A privacy safeguarding framework for the smart grid,” in IEEE ICC, 2021.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.