A General Purpose Data and Query Privacy Preserving Protocol for Wireless Sensor Networks

Niki Hrovatin, Aleksandar Tošić, Michael Mrissa, and Jernej Vičič

***Abstract*— The large number of devices characterizing Wireless Sensor Networks (WSNs) provide the benefits of observing, tracking, and recording everything; nonetheless, the cumulative computing power of those devices is typically not utilized, and the few implementations taking advantage of it neglect privacy or are application-specific. This manuscript describes a privacy-preserving protocol that enables WSN nodes to jointly compute an arbitrary function without disclosing their own private inputs. The computation takes place at data source nodes, while computation instructions and intermediate results move across the network secured by cryptography. The protocol relies on the Onion Routing technique to provide uniformly distributed network traffic and confine the knowledge a foreign actor can gain from monitoring messages traveling the network. We show that the communication protocol is privacy-preserving against the external and internal attacker models, and we validate our protocol implementation using the NS3 network simulator.**

***Index Terms*— Wireless sensor network, privacy, onion routing, distributed computing, data aggregation.**

I. Introduction

IN THE last decade, the cost reduction of sensor production and microelectronics has contributed to the development of large scale Wireless Sensor Networks (WSNs). WSNs are composed of dozens or hundreds of sensing nodes interlinked via radio signaling and meant to be easily deployed, self-configurable, and low cost. Moreover, nowadays, WSN nodes are powerful enough to store a short history of sensed data and process it. Therefore, the edge computing paradigm, which consists in moving computations as close as possible to data sources applies also to WSNs. Information fusion [1], declarative query processing [2], and inference [3] were all considered for in-network execution.

Manuscript received 24 October 2022; revised 30 May 2023 and 12 July 2023; accepted 24 July 2023. Date of publication 1 August 2023; date of current version 11 August 2023. This work was supported in part by the European Commission for funding the InnoRenew Project through the Horizon2020 Widespread-Teaming Program and the Republic of Slovenia (Investment Funding of the Republic of Slovenia and the European Regional Development Fund) under Grant 739574 and in part by the Slovenian Research Agency ARRS under Project J2-2504. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Edgar Weippl. *(Corresponding author: Niki Hrovatin.)*

Niki Hrovatin, Aleksandar Tošić, and Michael Mrissa are with the InnoRenew CoE, 6310 Izola, Slovenia, and also with the Faculty of Mathematics, Natural Sciences and Information Technologies, University of Primorska, 6000 Koper, Slovenia (e-mail: niki.hrovatin@famnit.upr.si).

Jernej Vičič is with the Research Centre of the Slovenian Academy of Sciences and Arts, The Fran Ramovš Institute, 1000 Ljubljana, Slovenia, and also with the Faculty of Mathematics, Natural Sciences and Information Technologies, University of Primorska, 6000 Koper, Slovenia.

Digital Object Identifier 10.1109/TIFS.2023.3300524

However, in a typical WSN, data is moving through a wireless multi-hop network without infrastructure, nodes composing a WSN are generally low-cost and resource-constrained devices; therefore, it is challenging to secure the network and guarantee adequate privacy [4]. Moreover, in-network processing tasks require that several WSN nodes contribute with their sensed data and processing power to the joint evaluation of a function; therefore, the privacy of nodes contributing to the processing is also threatened by other nodes participating. Research to date in the WSN field has focused on providing confidentiality by transforming the data using cryptography [5] or other obfuscation techniques [6] and allowing several nodes to operate over the transformed data without the possibility of extracting the private data from other nodes involved in the joint computation. This approach led to the development of several solutions [7]; nonetheless, in solutions considered practical for WSNs, the technique used to conceal data restricts the set of operations applicable to the transformed data. There is some evidence that anonymous communication could provide privacy for joint computation tasks [8]; however, due to the high privacy requirements of some applications and the data being typically self-descriptive, this category of approaches has been neglected.

In this manuscript, instead of allowing a set of WSN nodes to evaluate a function over obfuscated data, we exploit anonymity to conceal nodes involved in the computation, establishing a strong separation between context and data. The WSN is a system of devices sensing features of the surrounding environment. The data collected by sensor nodes describe events or changes; the sensor node's location reveals where the phenomena occurred. Therefore, the WSN data loses its descriptive utility without enough contextual information. To achieve the separation between data and context, we design a novel communication protocol inspired from the Onion Routing [9] protocol that allows only specific nodes to participate in the joint evaluation of a function while concealing them among a set of nodes performing the onion routing operation without accessing the computation. Nodes participating in the joint computation receive a message enclosing computation instructions and a partial result. Privacy is secured as the technique conceals the context of the partial result from participating sensor nodes. Although the data, represented by the partial result, can be observed by other nodes involved in the computation, the context – such as information on the number, identity, and location of nodes contributing to the partial result – remains hidden. Therefore, without access to key contextual information, the partial result loses its significance for all nodes obtaining it, except for the message origin, which defined the message path and, as such, inherently possesses knowledge of the context.

Throughout the manuscript, we are looking into the specific case of indoor air quality monitoring since it receives an increasing interest as it contributes to reducing the environmental impact of buildings, to improving building occupants' well-being, and to enhancing future building design [10], [11]. WSN deployed for indoor air quality monitoring relies on sensors collecting data about temperature, CO_2, VOC, PM, relative humidity, etc.

Indeed, a large WSN relying on high quality transducers can reach prohibitive costs. Air quality transducers are particularly expensive, and low cost options are raising interest [12]. A common practice to reduce the overall WSN cost is the dense deployment of nodes equipped with low-cost transducers. Reading quality is then improved by aggregating data from multiple nodes. However, in typical WSN implementations, the data is aggregated only at network endpoints resulting in high communication overhead since each node needs to report its sensed value and missing the opportunity to take advantage of the nodes' computing capabilities when conveying the whole processing load on the end system.

There are several techniques for aggregating sensor readings in-network as data moves towards the sink nodes [13]. These techniques also provide privacy preservation focusing on data confidentiality and are designed to operate in periodic reporting settings. Whereas if these techniques are applied in the on-demand setting to retrieve data from the building's individual locations, the resulting network traffic could be informative. In particular, wireless communications use the radio frequency spectrum to broadcast signals over the air; transmitted signals can be intercepted, then analyzed and backtracked to illicitly acquire information. Several studies focus on this complication considering location privacy protection [14] to conceal the location of sink nodes or detected events in WSNs deployed in unattended environments.

However, the adoption of WSNs for indoor monitoring is increasing [15], [16], [17], [18], raising the need for further privacy measures against traffic analysis when aggregating data in-network in on-demand settings. Traffic analysis attacks [19] are of particular concern in building monitoring. These attacks allow adversaries to remain unnoticeable, simply listening to network traffic, and extracting features like message size, frequency, processing time, etc. By associating these features with facts or secrets, machine learning techniques can infer important details about the monitored environment, which could potentially lead to the compromise of building security.

This context motivates the need to combine privacy preservation and distributed data computing on WSN nodes to employ node's computing capabilities without revealing sensor readings or sensitive contextual information.

In this paper, we describe the following contributions:

1) We present a communication protocol that allows WSN nodes to jointly compute an arbitrary function over their inputs while ensuring the privacy of inputs and operating without revealing significant contextual information.
2) The communication protocol is based on a novel use of the Onion Routing [9] technique.
3) We provide privacy preservation analysis showing that the communication protocol is secure against the external and internal attacker models.
4) We provide results of the privacy-preserving protocol simulated using the NS3 simulator [20].

The rest of this paper is organized as follows: Section II gives a brief overview of the privacy preserving protocol. Section III reviews related work and highlights the originality of our solution. Section IV details the general-purpose data and query privacy preserving protocol. Section V gives privacy preservation analyses. Section VI presents results of the privacy preserving communication protocol simulated using the network simulator NS3. Section VIII concludes the manuscript and gives guidelines for future work.

II. Solution Overview

This manuscript describes a communication protocol based on the Onion Routing [9] technique for anonymous communication over a computer network. We similarly employ messages structured into encryption layers, such that a layer can be decrypted only by the targeted node revealing an inner encryption layer addressed to another node in the network. Therefore, message decryption is carried out gradually by leading the layered message across WSN nodes following the precise order given at message construction. Encryption layers are not enclosing only the inner layer, but also additional secret information revealed only to the node decrypting that layer. Path details and encryption keys are in this way conveyed to in-path nodes. Path details are delivered in encryption layers to not disclose the whole message path, such that a node receiving the message can identify only the previous sender and the next receiver of the message. Encryption key pairs, however, are delivered only to a subset of nodes in the message path. Moreover, and differently from the traditional onion routing [9], encryption keys are not used to establish an anonymous channel. Instead, encryption keys give access to the payload accompanying the layered object. Please note that pairs of symmetric encryption keys include distinct keys; however, pairs are chained through layers of the layered object, as can be seen from Fig. 1. Therefore, nodes in the message path serve as the anonymity set[1] for nodes accessing the payload since each node in the message path could potentially receive symmetric encryption keys from the decryption of the layered object. Consequently, the identity of nodes accessing the payload remains concealed to nodes receiving the message.

In this paper, the WSN acts as a service. Authorized users construct queries and issue them to WSN gateway nodes. We refer to a query as the message composed of a head

[1]Based on the definition given by Pfitzmann and Köhntopp [21], the anonymity set is the set of subjects that might cause an action. In our protocol, the anonymity set is the set of nodes deciphering a layer of a layered object. If layer decryption reveals encryption key pairs to a node, then the node executes the computation; the action.

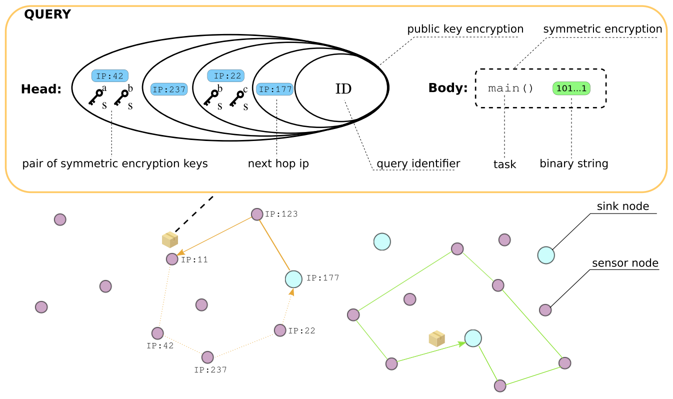

Fig. 1. Representation of the query, made of the head and the body. Notice that symmetric encryption keys are not enclosed in each layer of the query head. The figure displays the query's path forming a circuit.

consisting of the previously described layered object made of public-key encryption layers and a body consisting of the pair <*task, binary string*> as shown in Fig. 1. We refer to a task as computation instructions specifying the operations to be performed by sensor nodes. The binary string is of fixed size and stores task execution results. Each query head is constructed to lead the query over a path closing a circuit and delivering symmetric encryption key pairs to specific nodes in the query path allowing only them to access the query body content, execute the task and store results to the binary string. Other nodes in the query path not receiving symmetric encryption keys from query head decryption cannot access the query body content and only forward the query to the next-hop node. The query path closes a circuit, conveying back the results of the joint computation.

III. Related Work

Many systems were developed to preserve users' privacy while communicating on large public networks like the Internet. Several solutions originate from the work of *Chaum* [22] on mixnet. Mixnet-based schemes rely on a set of mix servers that receive encrypted messages, and after a sufficiently large amount of time, messages are re-ordered and released in batches to hide the correspondence between sender and receiver. Onion Routing [9] is a solution to preserve users' anonymity while avoiding latency introduced by mix servers. The solution relies on source routing using multiple encryption layers to route a message through a set of at least three routing servers (onion routers) to create an anonymous connection. Source routing [23] is a technique to route a message through a set of nodes by encoding path information in the message. In onion routing, path information is encrypted in each encryption layer of the message, and when the message is routed through onion routers, at each hop, a layer of encryption is removed from the message revealing the next hop. Therefore, no one of the actors involved in the communication will know the whole message path apart from the source of the message. In TOR [24] the second-generation onion routing, the message path is similarly protected by encryption layers, but the anonymous connection is established incrementally using key exchange schemes. Although the mentioned solutions effectively provide anonymous communication between two parties, they rely on the background traffic of large networks. Furthermore, the mentioned solutions protect the logical location of communicating parties (IP-address), while in WSNs, the privacy of communicating nodes can also be disclosed by observing the physical wireless communication. Despite the computational requirements and potential eavesdropping threats, the application of onion routing in WSNs has been proposed several times in the literature [25], [26], [27]. The paper [27] proposes an onion routing based mechanism for MQTT protocol communications, leveraging dynamic broker bridging to enable smart devices to subscribe and publish messages anonymously. In [26], a trust-based secure directed diffusion routing protocol for WSNs is proposed. The protocol leverages the onion routing for secure and anonymous end-to-end data transmission.

The literature review conducted by *Li et al.* [7] divides the privacy problem in WSNs into data privacy and context privacy. Data privacy is achieved if a communication protocol does not leak the collected data to external and internal adversaries. External adversaries eavesdrop on wireless communications, while internal adversaries have knowledge of some encryption keys used in the sensor network. In contrast, context privacy relates to communication traffic characteristics, as this can reveal insights over activities in the monitored environment. The paper [28] highlights the importance of concealing the sensor technology equipped on smart home devices, and the authors describe a technique for the periodic change of device identity that protects the network against external adversaries. Location privacy protection was extensively studied in event-driven WSNs using various routing strategies [14] aiming to conceal the location of data source nodes and sink nodes. The location of data source nodes can reveal insights over events detected by the WSN. Whereas keeping secret the location of sink nodes precludes the attacker from physically destroying sink nodes, which are of central importance for the correct functioning of the WSN. However, applying location privacy protection schemes in our scenario is not adequate since we aim to aggregate data from multiple nodes that may be closely located, and routing strategies designed to anonymize source and destination might disclose the region of interest.

The straightforward privacy-preserving approach to retrieve data from a region of interest in a WSN was highlighted by *Carbunar et al.* [29], and consists in gathering data from all sensor nodes in the network and then keep readings only from the sensor nodes of interest. Although effective, this approach is highly inefficient due to the multi-hop routing in WSNs requiring data to be relayed several times to reach the sink node. The efficiency of this approach was improved in [30] using compressive sensing [31] and public-key homomorphic encryption [32]. Compressive sensing is applied to transform sensor readings in a vector of coefficients which is aggregated with other node vectors along the routing paths to the sink node; therefore, requiring a low communication overhead of $O(MN)$, M the size of the vector of coefficients. Homomorphic encryption ensures privacy when aggregating vectors of coefficients.

The solution efficiently gathers data from multiple sensor nodes. However, in WSNs, neighboring nodes often share the same monitored area, and data sensed from neighbor nodes is often correlated. Therefore, data gathering schemes collecting raw data gather many duplicated data, do not utilize sensor node computing capabilities to process data, and pose a substantial communication load on the network even if the data need involves a small subset of network nodes.

In-network data aggregation [33] could effectively reduce the network's communication overhead and employ sensor node processing capabilities by aggregating multiple sensor node readings along routing paths toward the sink node. Therefore, the whole network works as a distributed processing mechanism delivering the final aggregated value to the sink node. Privacy preserving data aggregation solutions ensure data privacy against external and internal attackers [34], [35], [36]. However, proposed techniques do not consider the triggering of a data aggregation process that affects only a subset of nodes in the network without disclosing participating nodes. In building monitoring, the retrieval of aggregated data from the whole network of sensors can approximate the air quality. However, obtaining data from individual building locations is imperative to give a granular assessment.

The field of secure multi-party computation is concerned with enabling a set of parties to jointly compute an arbitrary function without disclosing their inputs [37]. A considerable amount of literature has been published on this topic, and some schemes have been implemented [38]. However, the secure multi-party computation fundamental protocols are resource demanding, and current implementations supporting arbitrary computation are not yet adequate for WSNs.

According to *Carbunar et al.* [29] a privacy-preserving query mechanism in WSNs must hide from attackers the location and identity of queried sensor nodes but also the relationship between individual queries while maintaining an adequate trade-off between privacy and efficiency. *Carbunar et al.* addressed query privacy needs with a WSN that acts as a service accessible through dedicated servers. The proposed solution hides query details from servers that provide access to the WSN. The WSN is mapped into regions and queries target aggregator nodes of individual regions. Query privacy is assured using source routing and by hiding a query constructed by the client with additional bogus queries targeting different regions of the WSN.

De Cristofaro et al. [25] proposed a privacy-preserving solution to retrieve WSN data without disclosing the identity of data source nodes to the network owner or attackers. The solution relies on source routing [23] using the onion routing to hide the query path and symmetric encryption to provide data privacy and data integrity. However, the proposed solution allows only retrieving individual sensor node readings without the possibility of in-network processing.

We state that our solution intrinsically follows the work on query privacy started by *Carbunar et al.* and *De Cristofaro et al.*. The originality of our proposed solution lies in the unique use of the onion routing technique to conceal nodes participating in the joint computation that takes place in-situ. Therefore, without aggregator nodes. Since aggregator nodes are gathering data from multiple sensor nodes, they are an appealing target for attackers and a point of failure for the network.

To the best of our knowledge, this is the first solution that allows the in-network joint computation of arbitrary functions in WSNs without aggregator nodes and keeps the participating nodes and their inputs private.

IV. Definition of the Privacy-Preserving Protocol

A. The WSN Model

Throughout the manuscript, we consider a WSN as a wireless multi-hop network consisting of two types of nodes. The majority are nodes equipped with sensing technology; we refer to these nodes as sensor nodes. The other type of node is named sink node, which purpose is to act as a gateway to external systems. The WSN relies on a routing protocol for multi-hop wireless networks, and sensor nodes in the WSN are configured at deployment with a static IP address and a public-private key pair. Sink nodes maintain a registry holding the following information of sensor nodes: IP address, public key, sensed physical quantities, and location of the building in which the sensor node is positioned (e.g., *room237*). The sink node registry can be accessed by authorized users willing to query the WSN. For simplicity, we refer to an authorized user as the user.

B. The Privacy-Preserving Communication Protocol

The network operates following an on-demand model: the user first access the sink node registry to obtain information about the WSN nodes, then constructs one or multiple queries as explained in Section IV-B.2; each query consists of a head and a body.

- **Head:** an onion-like structure made of encryption layers, the head is of fixed size L_H bytes. Layer decryption reveals the next-hop address (IP) or the next-hop address and a pair of symmetric encryption keys.
- **Body:** consists of t the task and w a fixed-size binary string used to transport the task execution results back to the user that issued the query. The query body is encrypted using symmetric encryption and is of fixed size L_B bytes.

Queries issued to the network follow the query path encoded in the query head, query processing at sensor nodes is explained in Section IV-B.3, and it branches based on the case that query head decryption reveals symmetric encryption keys. We refer to *decoy nodes* as the sensor nodes in the query path that do not receive symmetric encryption keys and do not participate in the joint computation and to *target nodes* as sensor nodes in the query path that receive symmetric encryption keys; thus, they can decipher the query body and participate in the joint computation. All queries travel a path forming a circuit that ends at the sink node that was the entry point of the query.

1) Query Preparation: The client application syncs to a sink node, downloading the registry of WSN nodes, then the user describes a data need expressing the request R_{eq}

consisting of the operation and its target. The operation is described by giving the task t, and the operation target by specifying τ, a set of one or multiple locations of the WSN.

Although the proposed solution allows conveying arbitrary computation instructions to sensor network nodes, the set of supported operations is bounded by the design of our protocol. Each query traverse several sensor nodes, and at each target node, the operation t is processed, acquiring input from sensors equipped on the sensor node and from w the binary string that carries results of t processed on the previous target node. Therefore, the solution is supporting operations that produce a partial result while feeding on the input of sensor values and the previous partial result. Between the common supporting data aggregation operations, we could list average, sum, max but also variance and standard deviation since in [39] was shown how to compute them as additive aggregation.[2] Moreover, our solution allows to pose conditions on the data to be retrieved, like the exceeding of a threshold value or past sensed events. Conditions can be posed not only on the physical quantity to retrieve but also on the state of other sensor technology equipped onto sensor nodes.

Algorithm 1 Query Path Selection

```
Require: $U$, $Q$
Ensure: $S$, $K$, $e_F$, $e_L$
  procedure RANDOM
      return a float from an uniform distribution bounded by (0,1)
  procedure PICKDECOY
      Chose randomly $s \in U \setminus (Q \cup B)$, add $s$ to $B$
      return $s$
  procedure PICKTARGET
      Chose randomly $s \in Q$, remove $s$ from $Q$, add $s$ to $B$
      return $s$
  procedure GENERATESYMKEY
      return a valid symmetric encryption key
 Compute:
 $B, S, K = \emptyset$
 $e_F$ =GENERATESYMKEY( )
 $k = null, e_L = e_F$
 $i, t = 1$
 $l = min(\|Q\|, \phi)$
 while $i \leq l$
     $t = \lceil$RANDOM( ) $* (n-1)\rceil$
     if S[t] == null
         $S[t]$ = PICKTARGET( )
         $i++$
     end if
 end while
 $i = 1$
 while $i \leq n$
     if S[i] != null
        k = ($e_L$,GENERATESYMKEY( ))
        $e_L$ = k[2]
     else
        S[i] = PICKDECOY( )
        k = null
     end if
     $K[i] = k$
     $i++$
 end while
```

A request R_{eq} specified by the user is processed as follows: t and τ are used to generate $Q \subset U$, the set of sensor nodes that are targets of the request. U is the set of all sensor nodes in the network. The set Q is generated by selecting sensor nodes from the set U that meet the location detailed in τ and sensed physical quantities required in t. The set P is then generated with the algorithm 1 repeated until $Q = \emptyset$. Algorithm 1 will eventually empty the set Q, since every call of the function *pickTarget()* removes a target from Q and inserts it in the query path. Each run of the algorithm will generate a query definition detailed by the tuple (S, K, e_F, e_L, id).

- $S = \langle s_1, \ldots, s_n \rangle$ a list consisting of sensor nodes belonging to the set U. The list S defines the query path.

[2]In additive aggregation, a sensor node sums its sensed value with a received partial result, and forwards the sum to the next sensor node.

- $K = \langle k_1, \ldots, k_n \rangle$ a list of elements k_i, where $k_i = (e_a, e_b)$ if s_i is a target node, otherwise s_i is a decoy node and $k_i = null$. (k_i and s_i the i-th elements of their respective list K and S, and (e_a, e_b) a pair of not equal symmetric encryption keys.) Moreover, encryption keys in the set K are arranged as follows: if s_i is a target node and s_j is the next target node in the query path, then $k_i = (e_a, e_b)$ and $k_j = (e_b, e_c)$.
- e_F the first symmetric key. If s_i is the first target node in the query path S, then $k_i = (e_F, e_x)$.
- e_L the last symmetric key. If s_i is the last target node in the query path S, then $k_i = (e_y, e_L)$.
- id a string of bits serving as the query identifier.

By our solution design, the query path length is a fixed network parameter; therefore, each query definition generated using algorithm 1 will include n nodes in its path. Algorithm 1 iteratively generates a query definition using two `while` loops. The first loop will inserts ϕ target nodes at random positions in the query path. Uncertainty is introduced to prevent queries from having a predictable disposition of target and decoy nodes. Since the last node in the query path can identify its function of being the node that will forward the query back to the sink node, by algorithm 1 this node is always a decoy node. The second loop will fill the query path with randomly chosen decoy nodes.

Algorithm 1 also outputs e_F and e_L. e_F is the first symmetric encryption key assigned to the first target node in the query path, this key is used to apply the first encryption layer on the query body. e_L is the second symmetric encryption key assigned to the last target node in the query path, this key is used by the client to decrypt the query body, and retrieve the query result.

The key e_L is coupled to the query identifier id and the pair is mapped into π the set of recovery rules. Since a request will inquire data from a large set of nodes, and each query can include at most ϕ target nodes in its path, the typical request will be accomplished by issuing multiple queries. The set of recovery rules π holds identifiers of queries issued to accomplish one request, and is used to recover the final request result.

To summarize, the request R_{eq} is translated into P – a non empty set of tuples of cardinality $\|P\| = \left\lceil \frac{\|Q\|}{\phi} \right\rceil$, a task t, and π a set of recovery rules. For each tuple $(S, K, e_F, e_L, id) \in P$ a query is constructed as explained in Section IV-B.2.

2) Query Construction: In this section, we describe how the client application converts a query definition detailed by the tuple (S, K, e_F, e_L, id), and a task t into a query consisting of the *head* and the *body*. The $\varepsilon(\cdot)$ denotes the encryption operation using public-key cryptography, and the $E(\cdot)$ denotes the encryption operation using symmetric cryptography.

a) Head construction: The query head construction starts from OR_{n+1}, the innermost encryption layer, which securely delivers the query identifier back to the query issuer node. We refer to the query issuer as the sink node that dispatches the query to the network. The innermost onion layer is formed via encryption of the query identifier id and the padding p using the issuer's public key Y_{sink}. The padding p is introduced to maintain the head of fixed size if the query includes fewer than $\lfloor n/2 \rfloor$ target nodes. The following equation describes how to compute the innermost onion layer.

$$OR_{n+1} = \varepsilon_{Y_{sink}}(id, p)$$

Next, the client will compute the layer OR_n. This layer is closing the circuit, forwarding the query back to the query issuer. The layer is committed to s_n the last sensor node of the list S. The layer OR_n is computed like the layer OR_i, with the sole exception of including the sink node ip address ip_{sink} as the next-hop address. Therefore, we omit explaining layer OR_n construction, and we give layer OR_i construction in the following lines.

The layer OR_i addressed to the sensor node $s_i \in S$ (i as index of the i-th element in lists S, K and index of the i-th encryption layer of the query head) is computed in two distinct ways. *A:* following equation 1 if $k_i = null$, therefore node s_i is a decoy node. Layer OR_i is computed via encryption of the next hop ip address $ip_{s_{i+1}}$, and previous onion layer OR_{i+1} using the public key Y_{s_i} belonging to the sensor node s_i. *B:* equation 2 is applied if $k_i = (e_a, e_b)$, therefore node s_i is a target node. Layer OR_i is computed via encryption of the next hop ip address $ip_{s_{i+1}}$, the two symmetric encryption keys e_a and e_b, and the previous onion layer OR_{i+1} using the public key Y_{s_i} belonging to the sensor node s_i.

$$OR_i = \varepsilon_{Y_{s_i}}(ip_{s_{i+1}}, OR_{i+1}) \tag{1}$$

$$OR_i = \varepsilon_{Y_{s_i}}(ip_{s_{i+1}}, e_a, e_b, OR_{i+1}) \tag{2}$$

The layer construction repeats until the formation of OR_1, the head's first encryption layer, which is always of size L_H bytes.

b) Body construction: The query body B includes the task t and a fixed size binary string. Since the query body must be of fixed size L_B bytes, and the size of task t can vary, additional padding p of $L_t - size(t)$ bytes must be included into the query body. (L_t the maximum allowed task size in bytes, and $size()$ the function that returns the number of bytes of the given argument) Then the query body is constructed by encrypting the binary string w, the task t, and padding p using the symmetric encryption key e_F. Query body construction can be summarized using the following equation:

$$B = E_{e_F}(w, t, p)$$

Now the query is complete: OR_1 the query head and B the query body. The query coupled with the address of the first node in the query path $s_1 \in S$ and the query identifier is forwarded to the sink node. The sink node memorizes the query identifier and issues the query to s_1.

3) Query Processing: A sensor node $s_i \in S$ (i-th node in the list S) receiving the query performs the following steps: *query decryption*, *task execution*, and *query forwarding*.

a) Query decryption: The sensor node s_i decrypts the query head OR_i using its private key X_{s_i}. Query head decryption reveals the next hop IP address $ip_{s_{i+1}}$, the next onion layer OR_{i+1}, and if s_i is a target node, head decryption also reveals the pair of symmetric encryption keys (e_a, e_b).

b) Task execution: If the sensor node s_i received symmetric encryption keys from query head decryption, then s_i is a target node and will perform the following steps. Otherwise, if s_i is a decoy node, it will skip the following steps to perform the step *query forwarding*.

The sensor node s_i decrypts B (the query body) using the first symmetric key e_a revealing: the data-carrying string w, the task t, and the padding p. The task t gets executed sourcing input from w and sensors. A task is executed at most for Δ_t milliseconds otherwise, task execution is interrupted. Task execution returns w', a binary string holding task execution results. Then s_i constructs B' the query body consisting of the data carrying string w', the task t, and the padding p all encrypted using the second symmetric encryption key e_b. Therefore, B' is constructed as follows: $B' = E_{e_b}(w', t, p)$. Since the content of B' differs from B only in the binary string, but the binary string w' is of the same size of L_w bytes as w, then query body size is maintained uniform.

c) Query forwarding: Query head is reassembled by applying the technique for onion size uniformity introduced in [9]. The query head size is maintained fixed at L_H bytes by adding λ a padding of $size(OR_i) - size(OR_{i+1})$ random bytes at the end of the onion layer OR_{i+1}. Therefore, the query head is now consisting of $OR_{i+1} + \lambda$. After Δ_q milliseconds ($\Delta_t < \Delta_q$) from receiving the query, the sensor node will randomly choose f a `float`, and will wait for other $f \cdot \Delta_q$ milliseconds before forwarding the query to the next hop. Adequate bounding values should be selected for the randomly chosen r, e.g. $0 \leq f \leq 4$. After waiting the required time, the query made of the head $OR_{i+1} + \lambda$ and the body B' (or B if node s_i is a decoy node) is forwarded to the next hop s_{i+1} at the IP address $ip_{s_{i+1}}$.

4) Result Retrieval: Each query sent to a sink node to accomplish the request R_{eq} will follow a path forming a circuit that ends at the sink node that issued the query. The sink node decrypts the query head consisting of the onion layer OR_{n+1} revealing id the query identifier. The query identifier and the query body B are then forwarded to the client that sent the query to the sink node. The client uses the query identifier to find the corresponding symmetric encryption key e_L from the recovery rules π, and the data-carrying string w is obtained from the query body decryption.

When the client gathers the feedback of all queries issued to accomplish R_{eq}, it starts the recovery process of the request result. Query results are merged following recovery rules π to obtain the end result of the request R_{eq}.

C. Arbitrary Computation

In this section, we discuss the capability of our proposed approach to perform multiple computations to accomplish an arbitrary computational job. Section IV-B described how the client application is used to construct multiple queries for retrieving an aggregated value from the WSN. In this process, all the queries issued for obtaining the desired aggregate enclose all the same task. However, the approach can be extended to enable the client application to construct several queries enclosing different tasks and using the binary string to maintain a context for the task. By properly arranging

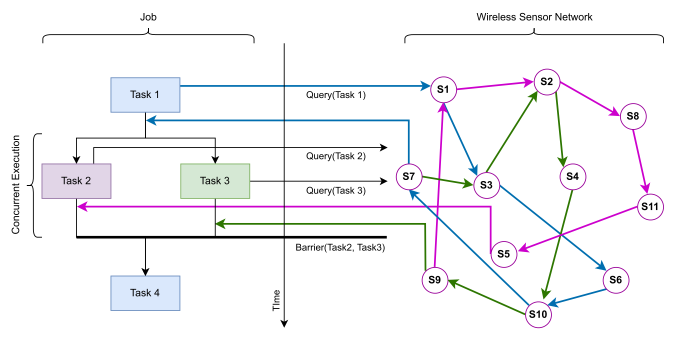

Fig. 2. Representation of a computational job divided into individual sub-tasks. Each sub-task independently executes a specific query within the WSN. The barrier synchronization manages task dependencies.

and merging the results from these queries, the client can accomplish the computational job.

To perform this, the client should first split a computational job into sub-tasks and identify those that can be executed independently and those dependent on the result of other tasks. Moreover, while splitting the computational job into sub-tasks, the client must consider the technique peculiarity of the sequential movement of the query from one node to another, which means that only the content of the query binary string and the local data of the node currently processing the query is available at each query processing step.

To manage the execution of these sub-tasks and their dependencies, the client can implement barrier synchronization. Barrier synchronization allows the client to wait for the completion of a set of queries before proceeding with the next set of queries. This ensures that any dependencies among the sub-tasks are properly managed and that the required results from previous queries are available before the subsequent queries are issued. The barrier synchronization of queries is shown in Fig. 2.

By using barrier synchronization at the client level, our approach can support the execution of complex computational jobs that involve multiple, interdependent sub-tasks. This extends the versatility of our technique, enabling the execution of more sophisticated computations across the WSN while still benefiting from privacy preservation.

D. Query Size Optimization and Query Authenticity

The technique described in this manuscript is based on queries consisting of several public-key encryption layers, and layers include encryption key material and path details. These queries are routed through a wireless multi-hop network and relayed several times. Therefore, it is important to minimize the query size to save network resources and improve the system's response time. Moreover, since queries are forwarded among sensor nodes, it is important to grant query authenticity allowing only authorized users to pose queries to sensor nodes and inhibit the injection of malicious queries. We define that a query is authentic if it was signed with an authorized private key. In the following, we describe how to optimize the query size and provide query authenticity using Elliptic Curve Cryptography (ECC) [40]. Several research [41], [42] suggest the adequacy of ECC cryptosystems for resource constrained devices, mainly due to the smaller key size compared to RSA.

To generate a digital signature we use the Edwards-curve Digital Signature Algorithm (EdDSA) [43]. The EdDSA signature of the message m consists of two values: R and s. The signature is computed using the signer's secret and public key, respectively an integer a and the point on curve $A = a \cdot G$. G is the base point on an elliptic curve defined over the finite field $\mathbb{F}_p$ of p elements with p prime. In EdDSA, R is obtained as $R = r \cdot G$, r a deterministic pseudorandom value, obtained from the hash of the last 32 bytes of a and the message to be signed. The value s is obtained from $s = r + HASH(R||A||message) \cdot a$. (We use $||$ to denote concatenation.) The signature verifier generates two values $v_1 = s \cdot G$, and $v_2 = R + A \cdot HASH(R||A||message)$. If $v_1 == v_2$, the signature checks.

To ensure authenticity, the user constructing the query should include a signature for each target node in the query path. For a target node with address IP_s, the message to be signed consists of $m = t||IP_s$. To prevent reply attacks and ensure that the query binary representation changes with each new query, the pseudorandom generation of the value r should be derived by hashing a, m, and a session ID. The Session ID should be appropriately generated to avoid the widely known security failure that lead to the compromise of the Sony PlayStation3.

The first trick to optimize the query size is that R can be reused for multiple ECC operations, which was shown to be secure in [44], and was also applied by *De Cristofaro et al.* in [25]. We emphasize that we use R for one signature and multiple ECC Diffie Hellman [45] operations. Specifically, we adopt the Diffie Hellman key encapsulation mechanism of the Elliptic Curve Integrated Encryption Scheme (ECIES) [46]. The ECIES allows deriving a secret ς from a random integer r and a public key. The same secret can be derived by the corresponding private key and the random integer. Therefore, by sharing the random integer, two parties derive the same secret. The random integer can be securely shared via its multiplication with the base point $R = r \cdot G$. Therefore, the value R used in the EdDSA signature can be reused in the ECIES to derive a shared secret. The derived shared secret ς is used in the ECIES as a key for applying a symmetric cipher over data. Moreover, since it is enough to verify the message authenticity at target nodes, the value R can be reused for the ECIES operation at decoy nodes.

The second optimization trick relates to the delivery of symmetric encryption keys to target nodes. One symmetric encryption key can be derived by hashing the shared secret ς obtained from ECIES decryption. A similar approach was applied in the original onion routing technique [47]. The second symmetric encryption key should be delivered in query head layers since keys are chained, as shown in Fig. 1. The delivered key should correspond to the value obtained by hashing the shared secret ς obtained by the next target node.

Therefore, the query head size can be summarised with equation 3, n the query path length, and $L_{\_}$ the size of the subscript argument.

$$L_R + (\frac{n}{2} \cdot (L_{IP}, L_R, L_s, L_{sym})) + (\frac{n}{2} \cdot (L_{IP})) \quad (3)$$

V. Privacy-Preservation Analysis

This section examines the communication protocol to identify vulnerabilities concerning privacy preservation. The literature on WSNs generally categorizes privacy concerns into two main classifications: external and internal privacy [7], [19], [25], [35], [36]. This paper aligns with this approach and utilizes the same terminology. External privacy is threatened by actors outside of the network listening to the wireless communication, while internal privacy is threatened by trusted participating sensor nodes of the WSN.

Throughout the analysis, we assume that the network implements layer 2 security, the guarantee of in-order data delivery using TCP, sink nodes cannot be compromised, and users do not collaborate with the attacker. For simplicity, we assume that messages are not fragmented.

A. External Privacy

To examine external privacy, we visualize a foreign actor that is monitoring the network traffic by eavesdropping on wireless communications. We dub this actor the external adversary. Eavesdropping is the intercepting and reading of messages by unintended receivers. Since the majority of wireless communications use the radio frequency spectrum to broadcast signals over the air, transmitted signals can be easily intercepted using adequate receiving equipment [48]. Event though the network implements layer 2 security, we assume that the external adversary is able to differentiate a transmission transferring query information from ordinary network management traffic by observing the transmission length.

Since layer 2 security protects transmissions using encryption and query size is maintained uniform throughout the query path, the only disclosed detail of an intercepted query is the effective transferring of the query from one node to another. The node receiver of the transmission is then processing the query (as explained in Section IV-B.3) or re-transmitting it to another node (routing in multi-hop networks). However, processing the query introduces a delay, missing if it is just re-transmitted. Therefore nodes processing the query can be identified; nonetheless, the external adversary cannot differentiate decoy nodes from target nodes since the query sojourn time at both kinds of nodes depends upon a randomly chosen `float`.

We now consider an external adversary whose monitoring range covers the whole WSN; therefore, it can intercept the whole wireless traffic generated by the WSN. Hence, the adversary can track a query sourcing from the sink node and moving through the network by monitoring its transmissions. However, normally, the WSN traffic is not populated by only one query, and the randomized nature of the query path will make various queries mix at nodes on their route. Even though the adversary violates security measures of the physical layer, security at the data link layer is changing data by encryption before each transmission. Furthermore, query size is maintained uniform; therefore, it is hard for an external adversary to track how the query transit through the network since the adversary cannot distinguish between queries.

B. Internal Privacy

An attacker that owns a subset of WSN nodes is commonly referred to as an internal adversary. Nodes owned by an internal adversary are participating trusted nodes of the WSN owning cryptographic keys to decrypt messages addressed to them. Section IV-D shows how to render void query injection attacks using digital signatures. Therefore, traffic analysis is the only attack that could compromise privacy concerning assumptions in Section V and the assumption of secure cryptographic primitives.

The internal adversary can take advantage of owned nodes to analyze traffic they receive and disclose information from un-compromised nodes of the WSN. Although we assumed that the network implements layer 2 security, only individual links are secured by such a solution, and nodes intermediate to routing paths can overhear messages passing through them. In the following, we will analyze under which circumstances an internal adversary is able to gain insights over other nodes in the network and when the data privacy of a sensor node is disclosed.

To simplify the analysis, we introduce the following notation expressing the operation of sensor nodes in relation to a query: D are decoy nodes, T are target node, A are nodes owned by the adversary intermediate to the routing path of the query, AD are decoy nodes owned by the adversary, and AT are target nodes owned by the adversary. In the following, we explain the implications of a transiting query over sensor nodes.

T and D are nodes that will process the query (as explained in Section IV-B.3). On these nodes, the adversary is trying to gain information. Nodes T are target nodes for the query, and after processing at T nodes, the redirected query is entirely changed by encryption. On the other hand, after query processing at D nodes, the redirected query has query head changed by encryption, but the query body remains unchanged.

A are nodes owned by the adversary that receive the query due to routing needs in wireless multi-hop networks. Therefore, A nodes receiving the query can observe the encrypted query head and query body. Moreover, the IP header reveals the address of the previous and next node processing the query.

AD nodes are owned by the adversary and are processing the query. However, AD nodes cannot access the query body. Therefore, AD nodes disclose only the IP address of the previous and the next node processing the query.

AT nodes are owned by the adversary and are processing the query as target nodes. Therefore, nodes AT can decipher the query head layer addressed to them, revealing the next-hop IP address and a pair of symmetric encryption keys. Thus, they can decipher the query body and learn the task and the binary string that carries the partial result. The internal adversary can examine the task and disclose the function to be computed; hence the adversary can identify the value carried in the binary string. We say identify since the adversary can recognize that the value is a sum, an extreme, etc. Although the internal

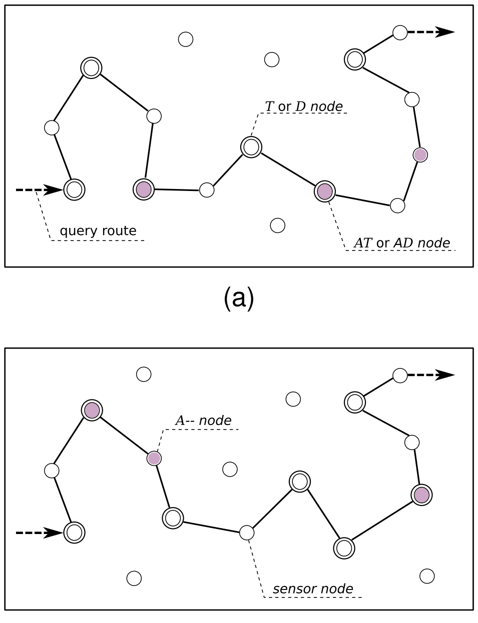

Fig. 3. (a) Route of the query that leads through one $T|D$ node confined between two nodes owned by the adversary. (b) Route of the query that leads through two or more $T|D$ nodes confined between two nodes owned by the adversary.

adversary can identify the value carried in the binary string, by owning a single node in the query path, the internal adversary cannot draw conclusions on the value or extract sensor node readings of other target nodes in the query path since it does not know which are target nodes and how many target nodes contributed to the partial result.

We now consider a query that transits through multiple nodes owned by the internal adversary. Hence the adversary can track sender and receiver IP enclosed in the IP header to partially reconstruct the query path and observe how the query body changes to gain information over other nodes in the query path. Note, we refer to the query path as the sequence of nodes on which the query is processed; on the other hand, we refer to the query route as the sequence of nodes that the query transits, including nodes in the query path and nodes that are forwarding the query due to routing needs in wireless multi-hop networks. To conduct this investigation we examine two cases where the query route leads through two nodes owned by the adversary: 1) two or more $T|D$ nodes confined between two nodes owned by the adversary; (_|_ used as exclusive OR) 2) one $T|D$ node confined between two nodes owned by the adversary.

Fig. 3 shows an example of the two cases. We consider these two cases since if the query route leads through more than two nodes owned by the adversary, the instance can be generalized to multiple of the aforementioned cases. Moreover, in Section V-B.3, we consider concerns of query exit and entry point.

1) Route of the Query That Leads Through Two or More $T|D$odes Confined Between Two Nodes Owned by the Adversary: We first look at the instance that nodes owned by the adversary are not both AT. Since the query is processing at two or more consecutive un-compromised nodes, IP information accompanying the query cannot be used to determine if both nodes owned by the adversary received the same query. Therefore, the adversary must rely solely on the query body to disclose meaningful information. Indeed, if nodes in the query path arranged between the two nodes owned by the adversary are all D, the inner encryption layer of the query body will remain unchanged. Therefore, the adversary can identify that both owned nodes received the same query and that all nodes processing the query between the owned nodes are D nodes. However, the internal adversary is not able to recognize the number of nodes in the query path between the two owned nodes since query sojourn time depends upon a randomly chosen `float`. On the other hand, if any node arranged between the two nodes owned by the adversary is a T node, then the query body is also changed by encryption, and the internal adversary cannot determine if both owned nodes are executing the same query.

We now consider that both nodes owned by the adversary are AT nodes. Then the adversary can decipher the query body at both owned nodes, and it should be possible to compare tasks and binary strings to recognize if both nodes received the same query. However, even though the adversary can recognize the value change of the binary string, he cannot identify which nodes processing the query are T nodes nor how many T nodes contributed to the value change.

2) Route of the Query That Leads Through One $T|D$ode Confined Between Two Nodes Owned by the Adversary: This disposition of nodes can be identified by the internal adversary as a transitive dependency of sender and receiver IP addresses from the IP packet header (e.g. $IP_{owned1} \rightarrow IP_i, IP_i \rightarrow IP_{owned2}$). Therefore, we assume that given the transitive relation of IP addresses, the adversary deduces that the two owned nodes are processing the same query, even if the query was changed by encryption and the randomly chosen query sojourn time does not ensure that it is the same query. Another query might be routed through $IP_i \rightarrow IP_{owned2}$ tricking the internal adversary of detecting the relation $IP_{owned1} \rightarrow IP_i, IP_i \rightarrow IP_{owned2}$.

Regardless of the aforementioned possibility of mixing queries, if the adversary identifies this particular disposition of nodes, he can examine the query received at the two owned nodes to gain insights over the node between them. We distinguish the following three cases where the adversary discloses different insights over the un-compromised node between the two owned nodes:

1) *Both nodes owned by the adversary are $A|AD$.* The adversary can compare the encrypted query body at both owned nodes to disclose if the un-compromised node is a T or D node for the received query.
2) *One node owned by the adversary is AT.* The adversary can recognize if the un-compromised node is a T or D node for the received query. Furthermore, if the un-compromised node is a T node, the adversary can observe the task to disclose the sensor technology equipped on the node.
3) *Both nodes owned by the adversary are AT.* In this case, the adversary can gain insights summarized in

the previous points. Additionally, if the un-compromised node is a T node, the adversary can compare the binary string state at the two owned nodes to extract data added by the un-compromised node.

3) Query Entry & Exit Point: We refer to the query entry point as the first sensor node in the query path and the query exit point as the last node in the query path since they both communicate with the sink node making them recognizable as such.

By our solution design, the first node in the query path can be a T node. Thus if the first and the second node in the query path are respectively T and AT, the adversary can disclose data privacy of the first node in the query path, however only if the adversary can identify that he owns the second node in the query path. Since the position of nodes in the query path can be identified only by tracking the query route from the source, this vulnerability can be exploited only if the adversary owns an A node in the query route from the sink node to the first node in the query path. Moreover, it is possible to avoid this vulnerability by setting an initial value to the data carrying string.

The query exit point or the last node in the query path is always a D node. We included this design choice since the last node in the query path can identify its position in the query from the next-hop IP address, which is the address of the sink node. If AT is the last node in the query path able to decipher the query body, the adversary could potentially infer over the number of T nodes that contributed to the partial result, since a query generally includes ϕ T nodes in the query path. However, the adversary can identify that he owns the last sensor node deciphering the query body only by tracking the query path to the sink node.

C. Data Leak Probability

The examination in Section V-B shows that under certain conditions, there is some probability that private sensor data leaks to the internal adversary. We use the term leak since the disclosure of private data depends on the randomized selection of nodes in the query path. The data leak occurs when the query path leads through three consecutive target nodes and the two outer nodes are owned by the adversary, then the adversary can compare the binary string state at the two owned nodes to disclose the data added by the inner node.

To determine the probability of a data leak during normal operation, we make the following assumptions: the selection of nodes forming the query path occurs using a uniform random generator, nodes in the query path do not repeat, and target nodes are randomly selected. Therefore in a WSN of s sensor nodes, the data leak probability depends on the query path length n, the number of target nodes in the query path ϕ, and on the number of sensor nodes owned by the adversary a. We count all the instances where selecting three target nodes, two of them are owned by the adversary; there are $n-2$ such

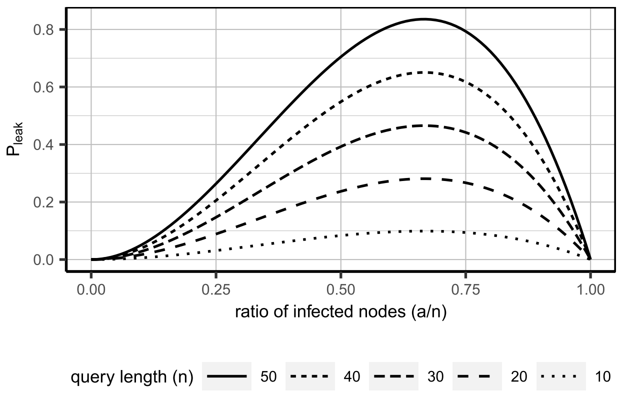

Fig. 4. The data leak probability for different query path lengths. ϕ is set to the default $\phi = n/2$.

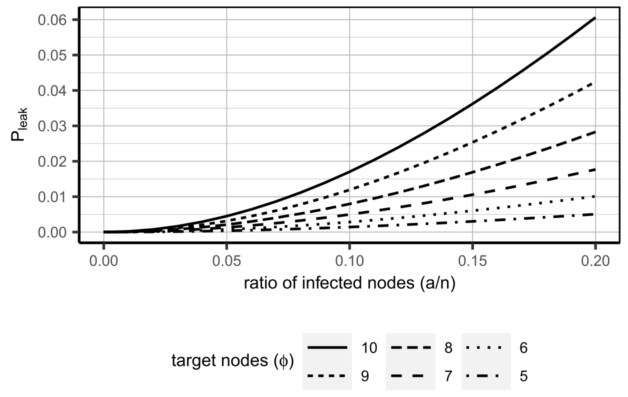

Fig. 5. The data leak probability at $n = 20$ and varying ϕ the number of target nodes in the query.

occurrences where the adversary does not own the inner node. The data leak probability is given in equation 5, as in (4), shown at the bottom of the page.

We emphasize that equation 5 has also been validated via simulation. We simulated the random query selection applying the same assumptions as for equation 5, and we counted the number of data leak occurrences. The simulation was repeated several times, and obtained values matched our analytical formula.

We plot the data leak probability distribution for different query path lengths in Fig. 4, and a zoomed version in Fig. 5 shows the decrease of the data leak probability when reducing the number of target nodes in the query path.

D. Active Attacks

The previous sections primarily focused on analyzing privacy-related attacks. However, WSNs also face a variety of additional threats that target data integrity and network availability. These types of attacks are known as active attacks, wherein adversaries actively interfere with the normal functioning of the WSN by manipulating data or disrupting

$$P_{leak} = \frac{a \cdot (n-\phi) \cdot (s-a) \cdot (n-\phi-1) \cdot (a-1) \cdot (n-\phi-2)(s-3)!}{s! \cdot n \cdot (n-1)} \tag{4}$$

network traffic. In this section, we will specifically discuss active attacks that could exploit our proposed technique to compromise the WSN while excluding attacks targeting other network layers, as they are well documented in the literature [4].

While the proposed technique offers certain security benefits, it does not fully protect against data integrity threats arising from physical environment manipulation or compromised nodes controlled by the internal adversary. Attackers may compromise data validity by interfering with sensor nodes' surroundings, such as artificially raising the temperature with a lighter. They can also undermine data integrity by altering query data using owned nodes. Specifically, target nodes processing the query have access to the query body. Although they cannot modify the task, as it is secured with query authenticity, as explained in Section IV-D, they can alter the binary string containing the aggregated data. Both scenarios can result in inaccurate data aggregation and potentially incorrect decision-making for the end user. However, such attacks on data integrity cannot specifically target individual nodes, as the proposed technique conceals the nodes processing the query. Consequently, these attacks would be inconsistent and more easily detectable.

In Section IV-D, we showed how to ensure query authenticity to prevent malicious query injection and replay attacks targeting data confidentiality. However, the technique we described only verifies query authenticity at the target nodes. Consequently, an internal adversary who knows the public keys associated with node addresses could forge a query that is only onion routed without providing access to the query body. The adversary's primary goal in doing so is to generate unnecessary traffic, leading to network congestion and disrupting the WSN's availability. An internal adversary could also execute an attack by dropping queries that are onion routed at compromised nodes. This attack is particularly effective in our proposed solution, as queries follow a fixed path and cannot bypass nodes that will onion-route them. As a result, the query would fail to reach its final destination, disrupting the data retrieval process and undermining the network's functionality.

While the described attacks can potentially impact the network's functioning, they share similarities with threats targeting other network layers. As such, these attacks can be mitigated by implementing intrusion detection systems [49].

E. Discussion of Privacy-Preservation Analyses

In Section V-A, we provide evidence that the solution withstands eavesdropping attacks since queries follow all the same circuit-like patterns while paths are randomized, transmitted queries are indistinguishable by encryption and uniform query size, and queries are mixing while transiting the WSN. Furthermore, external actors observing the wireless communication cannot disclose the nodes' target of the query since query forwarding time is decoupled from query execution.

In Section V-B we showed that the proposed protocol preserves query privacy by constraining information of the query path. Nodes receiving the query that can decipher the query body can learn about the task. However, without knowledge of the identities of other nodes involved in the joint computation, the adversary cannot infer information other than those revealed by the task.

Further analysis has shown that an attacker owning a portion of network nodes could possibly disclose insights about non-compromised sensor nodes and even threaten data privacy in the WSN. However, assuming secure cryptographic primitives, the adversary can increase the odds of privacy disclosures only by increasing the number of owned nodes in the network. Taking control over a sensor node is generally hard since it requires physical access to the node [50]. Moreover, due to the randomized selection of nodes forming the query path, the adversary cannot increase the data leak probability for a specific node in the network. In Section V-C, we expressed the data leak probability, and it was shown that it depends on the query path length, the number of target nodes in the query, and the number of nodes owned by the adversary. Therefore, the data leak probability can be adjusted based on the application's security requirements. Additionally, by applying the technique for query size optimization described in Section IV-D, adding a decoy node to the query path increases the query size only by the size of storing the node address; therefore, higher privacy requirements will result in a small decrease in efficiency.

In Section IV-D we explained how to use digital signatures to grant query authenticity. Here we want to emphasize that the described technique does not invalidate the security analyses in Section V. The signatures in the query head are all diverse since each signature is generated by coupling the task and a target node address. Therefore, digital signatures do not uniquely identify the query. Moreover, users could share signing keys if digital signatures are used only to verify query authenticity.

The privacy preservation analyses in Section V concern an attacker attacking the network. However, as discussed by *Carbunar et al.* [29] and *De Cristofaro et al.* [25] there are some circumstances where the WSN user wants to protect its privacy against the network owner. The solution of *De Cristofaro et al.* achieves a very high level of privacy; nonetheless, the technique allows only retrieving individual sensor node readings without the possibility of in-network processing. In the case of our technique, we incur the same issues as discussed by *Carbunar et al.*, requiring the user to generate multiple bogus queries to obfuscate the one of interest. The problem is extensively studied by *Carbunar et al.* in [29], taking into account spatial and temporal privacy.

We emphasize that protecting the user's privacy against the network owner is out of scope for this manuscript since here we focused on describing a technique for the joint computation of arbitrary functions on WSN nodes; moreover, in our solution, this problem could be addressed using code obfuscation techniques [51] to obfuscate the task and the carried data.

VI. Simulation Results

This section presents results of the privacy preserving communication protocol simulated using the NS3 [20], [52]. We examine the protocol scalability in a simulated WSN by

TABLE I
SUMMARY OF SIMULATION PARAMETERS AND CORRESPONDING VALUES

Param.	Value	Description
s	$50, 100, 200, 300, 400$	network size - number of sensor nodes
q	$5, 10, 15, , 20..., 50$	query path length
$topology$	GT, RDT	network topology grid / random disc
bs	$128, 256, 512, ..., 8192$	query body size (bytes)
$data\ rate$	$12\ Mbps$	transmission data rate at IEEE 802.11n

observing how querying is affected by the following independent variables: query path length (number of nodes on which the query will be processed), network size (number of nodes in the WSN), network topology, and query body size.

In order to quantify the response of independent variable adjustments, we meter the Round-Trip-Time (RTT) of queries. We define the RTT of a query, as the elapsed time between the issuing of the query from the sink node and the return of the issued query to the issuer node. Therefore, the RTT includes the sojourn time of the query at sensor nodes in the query path. However, Section IV-B.3 describes that the query sojourn time depends upon the application-specific parameter Δ_q and a randomly chosen `float` f. Therefore, we decided not measure the query randomized sojourn time since it introduces delays that are not dependent on the network. Moreover, by the law of large numbers [53], if choosing the value of f from a uniform random distribution bounded by the interval $[0, F]$, the average of the results obtained from a large number of trials will converge to the expected value of f. Therefore, the average cumulative sojourn time for a specific query path length n at an adequate Δ_q and F can be estimated using the following equation:

$$sojourn\_time = n \cdot \Delta_q \cdot \frac{F}{2} \tag{5}$$

A. Experimental Setup

The simulated WSN consists of one sink node and s sensor nodes. We consider two network topologies: the *grid topology* (GT) and the *random disc topology* (RDT). In the former, sensor nodes are deployed according to a grid structure; each sensor node is equidistant from the closest sensor nodes in cardinal directions. We set the distance between sensor nodes to $a = 60$ meters so that a sensor node is in the communication range of at most eight sensor nodes. In the latter, sensor nodes are randomly deployed on a disc-shaped plane of radius r_p. The radius r_p is obtained from $r_p = \sqrt{A/\pi}$, A being the sum of circular area's covered by s sensor nodes at radius $r_s = 35$ meters. Therefore, the average sensor node density of the network is maintained fixed at diverse s. Since in RDT sensor nodes are casually deployed on the target area, some sensor nodes could form independent network segments not connected to the network segment of which the sink node is a member. Therefore some sensor nodes might never be queried. In both network topologies, the sink node is deployed in the center of the WSN. We chose values of parameters $a = 60m$ and $r_s = 35m$, since networks of different topologies will have a near equal average node density: GT: $\frac{1}{60^2} = 0,277 * 10^{-3} nodes/m^2$, and RDT: $\frac{1}{35^2\pi} = 0,26*10^{-3} nodes/m^2$. Key simulation parameters are given in Table I.

The simulation implements the IEEE 802.11n standard for local wireless networks, operating in the 2.4 GHz band at the data rate of 12Mbps. The maximum segment size is set to the NS3 default 536 bytes. Each node in the WSN has installed the IP stack, and messages are transmitted over the TCP protocol.

However, the TCP was designed to function over low-error wired networks where the packet loss is usually the outcome of a network congestion [54]. Several studies are suggesting that the use of TCP in wireless multi-hop networks results in low throughput since packet loss due to transmission error and route discovery is handled using congestion avoidance and control [54], [55], [56]. Route discovery is performed by the routing protocol when searching for a route from sender to receiver. It is possible that discovering a route may take more time than the TCP retransmission timeout (RTO) [55]. The RTO is an internal timer of the TCP used to determine when a segment needs to be retransmitted. If the RTO elapses before receiving the acknowledgment of segment delivery, the segment is retransmitted, the RTO is increased using exponential backoff, and the TCP is adjusted for congestion. The minimum RTO value in the simulation is set to the default, 1 second. To avoid complications due to route discovery, we decided to use the Optimized Link State Routing Protocol (OLSR) [57], a proactive routing protocol so that routes are immediately available when needed. In proactive routing protocols, routes between each pair of nodes are determined at the network start-up and maintained with periodic updates.

Queries are constructed from the sink node by randomly selecting n nodes to include in the query path, n being the query path length. Query construction occurs using the technique for query size optimization ensuring query authenticity as explained in Section IV-D by using the elliptic curve Curve25519 [58] with key length 256-bit and the symmetric cipher Advanced Encryption Standard (AES) [59] at key length 128-bit. Therefore, recalling Section IV-D, R, s, and sym sizes are respectively 32 bytes, 32 bytes, and 16 bytes. To further reduce the query size, we use network addresses of 16-bits. The query body should be a multiple of the AES block size, and its size is adapted based on the experiment requirements.

Queries are issued from the sink node sequentially; after a query returns back to the sink node, the following query is issued. Nodes in the query path performing query processing maintain the uniform query size by adding padding. If the query does not reach the next-hop node in 100-seconds, the query is aborted, and a new query of equal parameters is issued from the sink node.

B. Experiment 1 – Remote Procedure Call

In this experiment, we consider the minimal size of the query body since this will give a better overview of how the network properties affect the RTT and since the query body size mainly depends on the computation instructions conveyed using the protocol.

The minimal query body size applies to the Remote Procedure Call (RPC) settings, where nodes of the WSN have encoded a set of functions, and the task t carried in the query

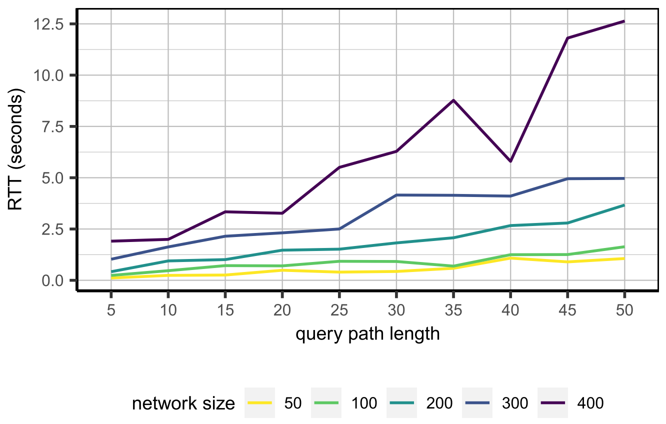

Fig. 6. The chart shows the average query RTT for the GT WSN at varying of the query path length and network size. Data from the experiment 1 in Section VI-B.

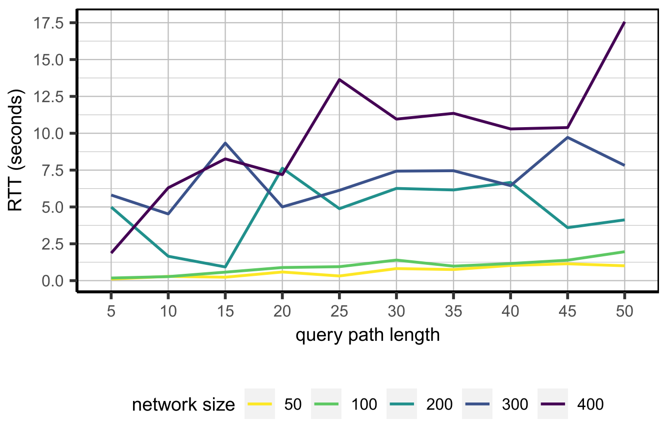

Fig. 7. The chart shows the average query RTT for the RDT WSN at varying of the query path length and network size. Data from the experiment 1 in Section VI-B.

body specifies which function to compute over sensor readings and w the data carrying string. We set the task t and the binary string w to 16 bytes.

A set of simulations was run for both GT and RDT at $s = \{50, 100, 200, 300, 400\}$. Each run executing 30 queries for each value of $n = \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$, the query path length. The obtained data is presented in Fig. 6, and Fig. 7. In the RDT simulations, the network segment, including the sink node, was connected to approximately 90% of all nodes. We observed some aborted queries in the RDT at the network size of 300 and 400 nodes. However, the events were all observed right after the simulation started; therefore, probably due to the OLSR routing requiring more time to converge. In the GT no aborted queries were detected.

C. Experiment 2 – Arbitrary Computation

In this experiment, we examine how the RTT is affected by the size of the query body. The large size of the query body is distinctive to an implementation of the communication protocol that delivers the function to be computed on sensor nodes within the query.

A set of simulations was run for the GT WSN at $s = \{50, 100, 200, 300, 400\}$. Each run executing 30 queries for the

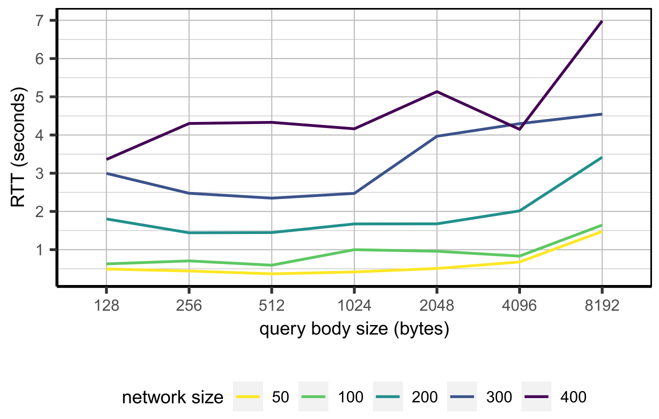

Fig. 8. The chart shows the average query RTT for the GT WSN with query path length set to $n = 20$, varying the query body size and network size. Data from the experiment 2 in Section VI-C.

query body size $bs = \{128, 256, 512, 1024, 2048, 4096, 8192\}$ bytes, at the query path length $n = 20$. The obtained data is presented in Fig. 8.

D. Discussion of Simulation Results

From Fig. 6, it is apparent that the query RTT is highly affected by the network size and the query path length. Furthermore, at larger network sizes the query path length has a higher effect on the query RTT. The high query RTT at large network sizes is due to the randomized selection of the query path that cause queries to cross nodes that may be very distant. Therefore, in a large network it should be practical to constrain the random selection of nodes to a network region that includes the nodes of interest but it is small enough to not incur in high query RTT.

The difference in query RTT between the GT and RDT can be observed in Fig. 6 and Fig. 7. In particular, it is possible to notice that the average query RTT at RDT is higher than at GT. Furthermore, the RDT data has a higher variance. Besides high variance of collected data, we also report a non-normal distribution of RTT measurements. A possible explanation for the non-normal distribution of RTT observation could be intrinsic to the TCP protocol and the RTO timer.

Experiment 2 considers the query RTT of queries with large query body sizes. Interestingly, in Fig. 8 can be seen that at selected parameters, the query body size does affect the average RTT of queries; however the increase is noticeable only for larger query body sizes (larger than 1024 bytes).

VII. Limitations

Although our proposed privacy-preserving protocol presents several advantages for WSNs, it also exhibits certain limitations that we discuss in this section.

A limitation related to our proposed technique that warrants further discussion is the use of layered messages constructed using public key encryption. Public key cryptography is well known for being more computationally intensive than symmetric key cryptography. However, public key encryption provides benefits that symmetric key encryption cannot, such as non-repudiation and no need for pre-shared keys. As a result, many studies apply public key encryption also in WSN settings [25], [60], [61]. The survey [62] provides an overview of public key cryptographic primitives adequate for WSNs, testing them on WSN devices. The suitability of 32-bit ARM-based microcontrollers for applications with significant cryptographic requirements is emphasized in [63] and [64], while [65] demonstrates the implementation of end-to-end integrity protection using Elliptic Curve Digital Signature Algorithm (ECDSA), further validating the efficacy of the 32-bit ARM architecture.

With regards to our protocol, it should be noted that even though messages consist of several public key encryption layers, the message creation process is performed outside the WSN on client applications. Consequently, the high overhead of applying multiple public key encryption layers does not load the WSN nodes. Upon receiving a query, WSN nodes perform either one or, at most, two public key operations. A single operation occurs if the query is only onion-routed through the node, while two operations take place when the node accesses the query body content and checks its authenticity. Therefore, our approach does not require many more public key operations than other techniques proposed for WSNs. Moreover, the computational load of decrypting query layers is uniformly distributed throughout the WSN.

Another limitation associated with the use of onion routing is the inability to change query routes after their creation. The underlying routing protocol can adapt to node failures by dynamically adjusting the routing path. For example, if a query is created to be processed at nodes A and C and must travel from A to C through a third node B, the route can be dynamically adjusted if node B fails, routing the query through an alternative path linking A and C. However, this cannot resolve issues arising from the fixed nature of onion routing. In cases where node C fails, the query will never reach its destination because path information can only be obtained by sequentially removing the encryption layers. Node C is the only node capable of removing the encryption layer addressed to it. If node C fails, no other node can remove that layer. In such situations, the network should adapt by flagging the failed sensor node or removing it from the registry held by sink nodes, making it ineligible for queries.

As observed in Section VI, our approach introduces a significant latency when processing a query, which impacts the real-time performance of the network. Furthermore, longer query path lengths result in longer query RTT. A potential solution to mitigate this issue is to send multiple smaller queries that can collectively obtain the same result, achieving reduced overall latency through parallel execution rather than sending a single query through a longer path. This approach allows for improved query response times while maintaining the desired privacy-preserving features.

It is important to note that the discussed limitations do not undermine the overall effectiveness of our privacy-preserving protocol in the context of WSNs. However, they provide insight into the trade-offs and challenges that need to be considered when implementing and deploying our proposed solution.

VIII. Conclusion and Further Work

This paper proposes a technique that enables WSN nodes to jointly compute a function in a privacy-preserving manner. We compare our proposal to the related work in the field, showing that, to the best of our knowledge, this is the first scheme that allows in-network joint computation of arbitrary functions in WSNs without aggregator nodes and without disclosing the nodes participating in the computation and their private inputs. We show that the scheme mitigates traffic analysis attacks; thereby making it adequate for indoor monitoring. As future work; we plan to implement and test our proposal in several buildings to collect and in-situ analyze air quality data. Additional future work includes investigating further adaptation of the proposed technique to IoT environments, the technique will be considered for training and evaluation of machine learning models and the federated learning paradigm could offer a valuable avenue for exploration. Moreover, the application of code obfuscation techniques or homomorphic encryption to the tasks conveyed to sensor nodes, would considerably enhance privacy preservation, warranting further exploration into the potential trade-offs of such an approach.

References

[1] E. F. Nakamura, A. A. F. Loureiro, and A. C. Frery, "Information fusion for wireless sensor networks: Methods, models, and classifications," *ACM Comput. Surv.*, vol. 39, no. 3, p. 9, Sep. 2007.

[2] F. Jabeen and S. Nawaz, "In-network wireless sensor network query processors: State of the art, challenges and future directions," *Inf. Fusion*, vol. 25, pp. 1–15, Sep. 2015.

[3] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 56–69, Jul. 2006.

[4] I. Tomic and J. A. McCann, "A survey of potential security issues in existing wireless sensor network protocols," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1910–1923, Dec. 2017.

[5] Y.-H. Lin, S.-Y. Chang, and H.-M. Sun, "CDAMA: Concealed data aggregation scheme for multiple applications in wireless sensor networks," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 7, pp. 1471–1483, Jul. 2013.

[6] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher, "PDA: Privacy-preserving data aggregation in wireless sensor networks," in *Proc. IEEE INFOCOM 26th Int. Conf. Comput. Commun.*, 2007, pp. 2045–2053.

[7] N. Li, N. Zhang, S. K. Das, and B. Thuraisingham, "Privacy preservation in wireless sensor networks: A state-of-the-art survey," *Ad Hoc Netw.*, vol. 7, no. 8, pp. 1501–1514, Nov. 2009.

[8] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Cryptography from anonymity," in *Proc. 47th Annu. IEEE Symp. Found. Comput. Sci. (FOCS)*, 2006, pp. 239–248.

[9] P. F. Syverson, D. M. Goldschlag, and M. G. Reed, "Anonymous connections and onion routing," in *Proc. IEEE Symp. Secur. Privacy*, May 1997, pp. 44–54.

[10] M. Mrissa et al., "Extending BIM for air quality monitoring," in *Construction Materials for a Sustainable Future*, vol. 1. Ljubljana, Slovenia: Slovenian National Building and Civil Engineering Institute, 2020, pp. 244–250.

[11] D. Clements-Croome, "Sustainable intelligent buildings for people: A review," *Intell. Buildings Int.*, vol. 3, no. 2, pp. 67–86, 2011.

[12] Y. Kang, L. Aye, T. D. Ngo, and J. Zhou, "Performance evaluation of low-cost air quality sensors: A review," *Sci. Total Environ.*, vol. 818, Apr. 2022, Art. no. 151769. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0048969721068455

[13] J. Xu, G. Yang, Z. Chen, and Q. Wang, "A survey on the privacy-preserving data aggregation in wireless sensor networks," *China Commun.*, vol. 12, no. 5, pp. 162–180, May 2015.

[14] J. Jiang, G. Han, H. Wang, and M. Guizani, "A survey on location privacy protection in wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 125, pp. 93–114, Jan. 2019.

[15] A. Maddumabandara, H. Leung, and M. Liu, "Experimental evaluation of indoor localization using wireless sensor networks," *IEEE Sensors J.*, vol. 15, no. 9, pp. 5228–5237, Sep. 2015.

[16] B. Andò, S. Baglio, and C. O. Lombardo, "RESIMA: An assistive paradigm to support weak people in indoor environments," *IEEE Trans. Instrum. Meas.*, vol. 63, no. 11, pp. 2522–2528, Nov. 2014.

[17] P. Zhou, G. Huang, L. Zhang, and K.-F. Tsang, "Wireless sensor network based monitoring system for a large-scale indoor space: Data process and supply air allocation optimization," *Energy Buildings*, vol. 103, pp. 365–374, Sep. 2015.

[18] J. Molka-Danielsen, P. Engelseth, and H. Wang, "Large scale integration of wireless sensor network technologies for air quality monitoring at a logistics shipping base," *J. Ind. Inf. Integr.*, vol. 10, pp. 20–28, Jun. 2018.

[19] J.-F. Raymond, "Traffic analysis: Protocols, attacks, design issues, and open problems," in *Designing Privacy Enhancing Technologies*. Cham, Switzerland: Springer, 2001, pp. 10–29.

[20] Nsnam. (Oct. 1, 2021). *Ns-3, a Discrete-Event Network Simulator for Internet Systems–Version 3.32*. [Online]. Available: https://www.nsnam.org/

[21] A. Pfitzmann and M. Hansen, "A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management," Version v0.34, Aug. 2010. [Online]. Available: http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf

[22] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–90, Feb. 1981.

[23] C. A. Sunshine, "Source routing in computer networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 7, no. 1, pp. 29–33, Jan. 1977.

[24] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," Naval Res. Lab, Washington, DC, USA, Tech. Rep. 0704-0188, 2004.

[25] E. D. Cristofaro, X. Ding, and G. Tsudik, "Privacy-preserving querying in sensor networks," in *Proc. 18th Int. Conf. Comput. Commun. Netw.*, Aug. 2009, pp. 1–6.

[26] X. Yu, F. Li, T. Li, N. Wu, H. Wang, and H. Zhou, "Trust-based secure directed diffusion routing protocol in WSN," *J. Ambient Intell. Humanized Comput.*, vol. 13, pp. 1405–1417, Nov. 2022.

[27] Y. Protskaya and L. Veltri, "Broker bridging mechanism for providing anonymity in MQTT," in *Proc. 10th Int. Conf. Netw. Future (NoF)*, Oct. 2019, pp. 110–113.

[28] Y. Alshboul, A. A. R. Bsoul, M. AL Zamil, and S. Samarah, "Cybersecurity of smart home systems: Sensor identity protection," *J. Netw. Syst. Manage.*, vol. 29, no. 3, pp. 1–27, Jul. 2021.

[29] B. Carbunar, Y. Yu, W. Shi, M. Pearce, and V. Vasudevan, "Query privacy in wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 6, no. 2, pp. 1–34, Feb. 2010.

[30] K. Xie et al., "An efficient privacy-preserving compressive data gathering scheme in WSNs," *Inf. Sci.*, vol. 390, pp. 82–94, Jun. 2017.

[31] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[32] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Cham, Switzerland: Springer, 1999, pp. 223–238.

[33] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proc. 6th Annu. Int. Conf. Mobile Comput. Netw.*, Aug. 2000, pp. 56–67.

[34] M. Conti, L. Zhang, S. Roy, R. Di Pietro, S. Jajodia, and L. V. Mancini, "Privacy-preserving robust data aggregation in wireless sensor networks," *Secur. Commun. Netw.*, vol. 2, no. 2, pp. 195–213, Mar. 2009.

[35] L. Zhang, H. Zhang, M. Conti, R. Di Pietro, S. Jajodia, and L. V. Mancini, "Preserving privacy against external and internal threats in WSN data aggregation," *Telecommun. Syst.*, vol. 52, no. 4, pp. 2163–2176, Apr. 2013.

[36] R. Bista and J.-W. Chang, "Privacy-preserving data aggregation protocols for wireless sensor networks: A survey," *Sensors*, vol. 10, no. 5, pp. 4577–4601, May 2010.

[37] C. Zhao et al., "Secure multi-party computation: Theory, practice and applications," *Inf. Sci.*, vol. 476, pp. 357–372, Feb. 2019.

[38] Y. Lindell, "Secure multiparty computation," *Commun. ACM*, vol. 64, no. 1, pp. 86–96, 2020.

[39] C. Castelluccia, E. Mykletun, and G. Tsudik, "Efficient aggregation of encrypted data in wireless sensor networks," in *Proc. 2nd Annu. Int. Conf. Mobile Ubiquitous Syst., Netw. Services*, 2005, pp. 109–117.

[40] V. S. Miller, "Use of elliptic curves in cryptography," in *Proc. Conf. Theory Appl. Cryptograph. Techn.* Cham, Switzerland: Springer, 1985, pp. 417–426.

[41] C. A. Lara-Nino, A. Diaz-Perez, and M. Morales-Sandoval, "Elliptic curve lightweight cryptography: A survey," *IEEE Access*, vol. 6, pp. 72514–72550, 2018.

[42] K. Maletsky, "RSA vs. ECC comparison for embedded systems," Microchip Technol. Inc., Chandler, AZ, USA, Tech. Rep. DS00003442A, 2020.

[43] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," *J. Cryptograph. Eng.*, vol. 2, no. 2, pp. 77–89, Sep. 2012.

[44] M. Bellare, A. Boldyreva, and J. Staddon, "Randomness re-use in multi-recipient encryption schemeas," in *Proc. Int. Workshop Public Key Cryptogr.* Cham, Switzerland: Springer, 2003, pp. 85–99.

[45] W. Diffie and M. E. Hellman, "New directions in cryptography," in *Secure Communications and Asymmetric Cryptosystems*. Evanston, IL, USA: Routledge, 2019, pp. 143–180.

[46] D. Galindo, S. Martin, and J. L. Villar, "Evaluating elliptic curve based KEMs in the light of pairings," Cryptol. ePrint Arch., Paper 2004/084, 2004. [Online]. Available: https://eprint.iacr.org/2004/084

[47] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Anonymous connections and onion routing," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 4, pp. 482–494, May 1998.

[48] B. Wu, J. Chen, J. Wu, and M. Cardei, "A survey of attacks and countermeasures in mobile ad hoc networks," in *Wireless Network Security*. Cham, Switzerland: Springer, 2007, pp. 103–135.

[49] B. B. Zarpelão, R. S Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *J. Netw. Comput. Appl.*, vol. 84, pp. 25–37, Apr. 2017.

[50] I. Butun, P. Osterberg, and H. Song, "Security of the Internet of Things: Vulnerabilities, attacks, and countermeasures," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 616–644, 1st Quart., 2020.

[51] B. Barak et al., "On the (IM) possibility of obfuscating programs," in *Proc. Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 2001, pp. 1–18.

[52] N. Hrovatin, A. Tošić, and J. Vičič, "PPWSim: Privacy preserving wireless sensor network simulator," *SoftwareX*, vol. 18, Jun. 2022, Art. no. 101067.

[53] P. L. Hsu and H. Robbins, "Complete convergence and the law of large numbers," *Proc. Nat. Acad. Sci. USA*, vol. 33, no. 2, pp. 25–31, Feb. 1947.

[54] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A feedback-based scheme for improving TCP performance in ad hoc wireless networks," *IEEE Pers. Commun.*, vol. 8, no. 1, pp. 34–39, 1st Quart., 2001.

[55] J. Liu and S. Singh, "ATCP: TCP for mobile ad hoc networks," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 7, pp. 1300–1315, Jul. 2001.

[56] C. Gomez, A. Arcia-Moret, and J. Crowcroft, "TCP in the Internet of Things: From ostracism to prominence," *IEEE Internet Comput.*, vol. 22, no. 1, pp. 29–41, Jan. 2018.

[57] T. Clausen et al., *Optimized Link State Routing Protocol (OLSR)*, document RFC 3626, INRIA, 2003.

[58] D. J. Bernstein, "Curve25519: New Diffie–Hellman speed records," in *Proc. Int. Workshop Public Key Cryptogr.* Cham, Switzerland: Springer, 2006, pp. 207–228.

[59] J. Daemen, "AES proposal: Rijndael," 1998. [Online]. Available: https://api.semanticscholar.org/CorpusID:17885291

[60] Y.-T. Tsou, C.-S. Lu, and S.-Y. Kuo, "SER: Secure and efficient retrieval for anonymous range query in wireless sensor networks," *Comput. Commun.*, vol. 108, pp. 1–16, Aug. 2017.

[61] C.-M. Chen, Y.-H. Lin, Y.-C. Lin, and H.-M. Sun, "RCDA: Recoverable concealed data aggregation for data integrity in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 4, pp. 727–734, Apr. 2012.

[62] K.-A. Shim, "A survey of public-key cryptographic primitives in wireless sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 577–601, 1st Quart., 2016.

[63] M. Sethi, J. Arkko, A. Keranen, and H. Back, "Practical considerations and implementation experiences in securing smart object networks," Internet Eng. Task Force (IETF), Fremont, CA, USA, Tech. Rep. RFC 8387, 2018.

[64] L. E. Kane, J. J. Chen, R. Thomas, V. Liu, and M. Mckague, "Security and performance in IoT: A balancing act," *IEEE Access*, vol. 8, pp. 121969–121986, 2020.

[65] J. Bauer, R. C. Staudemeyer, H. C. Pöhls, and A. Fragkiadakis, "ECDSA on things: IoT integrity protection in practise," in *Proc. Int. Conf. Inf. Commun. Secur.* Singapore: Springer, Dec. 2016, pp. 3–17.

Niki Hrovatin was born in Trieste, Italy, in 1994. He received the B.S. and M.S. degrees in computer science from the University of Primorska, Koper, Slovenia, in 2020, where he is currently pursuing the Ph.D. degree. He is also a Teaching Assistant with the University of Primorska and a Research Assistant with the InnoRenew CoE, Izola, Slovenia. His current research interests include sensor networks, machine learning, distributed systems, and blockchain.

Aleksandar Tošić is currently an Associate Professor with the University of Primorska, Koper, and a Researcher with the InnoRenew CoE. His main research interests include distributed systems, privacy and security, sensors, and distributed ledger technologies.

Michael Mrissa received the Ph.D. and HDR (accreditation to supervise research) degrees from Université Claude Bernard Lyon 1, France, in 2007 and 2014, respectively. He is currently a Full Professor in computer science with the University of Primorska and a Researcher with the InnoRenew CoE, Slovenia. He has published over 100 peer-reviewed papers in international conferences and journals and has been involved in numerous national and international projects. His research interests include privacy and security in distributed systems, sensor networks, and the Web of Things.

Jernej Vičič is currently an Associate Professor with the University of Primorska, Slovenia. He is also the Head of the Distributed Ledger Technologies and Language Technologies Laboratory. His research interests are mostly linked to the title of the laboratory.