ElliPS: A Privacy Preserving Scheme for Sensor Data Storage and Query

Nalin Subramanian, Ka Yang, Wensheng Zhang and Daji Qiao
Iowa State University, Ames, Iowa - 50011

***Abstract*—With in-network sensor data storage and query, storage nodes are responsible for storing the data collected by sensor nodes and answering queries from users. Thus, without proper protection for data types and user queries, compromise of storage nodes and/or sensor nodes may reveal sensitive information about the sensed environment as well as users' private interests and query patterns. In this paper, we explore trade-offs between privacy, computation overhead, communication overhead, network flexibility and network complexity, and propose ElliPS (Elliptic curve based Privacy Scheme) to provide joint protection on data type privacy and query privacy in the presence of sensor node compromise, storage node compromise, or under collusive attacks by compromised sensor nodes and storage nodes together. Extensive analysis and simulation are conducted to verify the security properties and efficiency of the proposed scheme.**

I. Introduction

A. Motivation and Contribution

In-network data storage and query architecture [1], [2] has been recognized as an effective solution for sensor data management. It is a two-tier architecture with a large number of regular sensor nodes and a smaller number of higher-storage-capacity *storage nodes*, which store data reported by regular sensor nodes and answer queries from users at the base station. To facilitate efficient query processing at storage nodes, each data report is usually associated with a *data type* (or *meta data*) to describe its characteristics, such as data generation time, data source ID, data source location, range of value, and so on, which provide valuable information and context about the data content. In fact, data content is meaningless if not considered jointly with the corresponding data type. On the other hand, user query is usually represented as a combination of data types of user's interest. In other words, data types facilitate the efficient query processing over raw data.

Storage nodes play a vital role for in-network data storage and query. Compromise of storage nodes may expose data types of stored data reports or data types of user queries, which may in turn reveal sensitive information about the sensed environment or users' private interests. Therefore, it is equally important to protect the data type privacy and query privacy in addition to the privacy of data contents. Protecting the privacy of data contents is protecting data confidentiality and has been well studied [3], [4]. However, there lacks systematic investigation on protecting data type privacy and query privacy under the in-network sensor data storage and query architecture.

A natural idea to conceal data types is to encrypt them. For example, sensor nodes may use the the secret keys they share with the base station to encrypt the contents and types of sensor data before sending to the storage nodes, and the base station may send queries in the form of encrypted data types. Since the secret keys are not known to the storage nodes, this scheme works fine if the storage node is the only node compromised in the network. However, if some sensor nodes are also compromised, they may collude with the compromised storage nodes to reveal the query privacy and to infer the data types of data reported by other innocent sensor nodes, which is discussed in detail in Section III-A1. In other words, such simple encryption schemes may not protect the data type privacy and query privacy effectively under collusive attacks by compromised sensor nodes and storage nodes.

Therefore, a more elaborated collusion-resilient privacy-preserving scheming is needed for in-network sensor data storage and query. However, higher computation and communication overhead, higher network complexity, less network flexibility, more time delay, etc., will be expected. In fact, for all kinds of networks, there is an intrinsic set of trade-offs between computation overhead, communication overhead, network complexity, network flexibility, etc., and privacy. Better performance in one aspect always trades off the performance of one or more other aspects. In the above example, to defend against the collusion between compromised sensors and storage nodes, intermediate nodes can be introduced to break the direct link between compromised sensors and storage nodes, which will increase the network complexity. In our work, by exploring the trade-offs between all these aspects of a wireless sensor network, we aim to find a solution for preserving privacy for in-network sensor data storage and query while satisfying the following requirements:

- *Contents and types of reported sensor data are concealed.*
- *User queries from the base station are concealed.*
- *Resilience to collusive attacks by compromised sensor nodes and storage nodes.*
- *Non-disruption to normal query processing.*
- *Low computation, communication and storage overheads.*

In this paper, we propose ElliPS (Elliptic curve based Privacy Scheme) as a solution to meet all the above requirements. ElliPS is developed based on novel applications of the following techniques:

- *Collaborative mix and function compounding.*
- *Delicate design and manipulated distribution of finite field and/or elliptic curve group-based polynomials.*

The key ideas of ElliPS are to use carefully-designed elliptic curve group-based polynomials to conceal data types of reported sensor data and user queries without disrupting normal query processing at storage nodes, and to introduce special *anonymizer nodes* between sensor nodes and storage nodes to break the potential collusion between compromised sensor nodes and storage nodes. In our scheme, we trade off computation overhead, communication overhead, network flexibility and network complexity, etc., for the privacy of data type and query within an acceptable range. Extensive analysis and simulation have been conducted to verify the security properties and efficiency of the proposed scheme.

B. Related Work

Using *data type* (also referred to as *keyword* or *attribute* in the literatures) to facilitate searching over encrypted data has been proposed and studied extensively for various types of data management systems [5]–[8]. In [5], [6], keywords are transformed in such a way that the privacy of keywords are preserved while transformed keywords may still be used to search over encrypted files. In [7], [8], advanced privacy-preserving schemes were proposed to allow more sophisticated multi-attribute search over encrypted files while preserving the privacy of these attributes. Since in these schemes network model has trustable data sources, they cannot be directly applied for data management in sensor networks where both storage nodes and sensor nodes may be compromised.

Query privacy in sensor networks has started receiving more attention in the past few years. In [9], query privacy of the client is protected from un-trusted servers. However, this work is limited to protect only the ID of the sensor node which the client may be interested in. The authors of [10] proposed a scheme to protect the privacy of sensor data and range queries from the base station, under the in-network data storage and query architecture. But it cannot deal with collusive attacks by compromised sensor nodes and storage nodes.

The concept of "anonymizer" has been used for Internet applications before. In [11], the authors proposed a system based on public key cryptography. It hides the correspondence between sender and receiver of an email by encrypting messages repeatedly with the public keys of a predefined set of mixes (i.e., anonymizers). A cryptography-based anonymous system was also used in [12]. In contrast, anonymizers in our scheme employ light-weight polynomial-based technique, and the purpose of having anonymizers in our scheme is to hide the untrustworthy data source from the storage nodes, which is different from previous works.

Our scheme is different from all the above schemes in that it protects the data type privacy and query privacy in the presence of compromised sensor nodes, compromised anonymizers or compromised storage nodes, or under collusive attacks launched by them together.

C. Organization

The rest of the paper is organized as follows. Sec. II presents the models and assumptions, Sec. III discusses why simple encryption schemes fail to counter collusive attacks on data type privacy and query privacy, and describes our proposed ElliPS scheme in detail and Sec. IV analyzes its security properties. Sec. V reports the performance analysis and evaluation results, Sec. VI discusses some related issues. Finally, Sec. VII concludes the paper.

II. Models and Assumptions

A. Network Assumptions

We consider a heterogeneous sensor network consisting of a large number of regular resource-constrained sensor nodes (some of them serve as anonymizer nodes), a small number of resource-rich storage nodes and a base station. Regular sensor nodes sense the environment, generate data and report them to one of the storage nodes. User queries the storage nodes via base station for the data of interest. Our current query model allows the users to forward their queries and receive responses through the base station. In the future we will extend the query model to include in-network users querying the storage nodes directly, where users may be compromisable and untrustworthy.

Each data report generated and transferred in the network has two parts: *data type* and *data content*. Queries for sensor data are based on data types. For example, consider a sensor network used for monitoring temperature. The content of each data report generated in this network includes a temperature reading and the geographic position and time instance at which the reading was obtained. The data type is one of the predefined ranges, e.g., $(-\infty, 0)$, $[0, 5)$, $\cdots$, and $[100, \infty)$, which is the range the temperature reading falls in. We assume all the data reporting sensors are preloaded with application specific data type knowledge by the network controller. Each query contains one or more of the predefined ranges to specify the set of data reports that match user's interest. The data type may contain more than one attributes. For instance, the data type in the afore-described example may also contain geographic range, temporal range, and so on. All these attributes can be used in a query to specify user's interest. In this paper, we assume that each data type contains only one attribute to simplify the presentation, though our proposed solution could be extended to multi-attribute data types without much difficulty.

B. Security Assumptions

In our system, we assume that the base station is trustable and non-compromisable. However, other nodes in the network, such as data reporting sensors, anonymizers and storage node, may be compromised. We consider the following attacks:

- *Outsider Attacks* – launched by the adversary who has not compromised any node in the network. By eavesdropping messages in transit, the adversary attempts to infer the types of data generated, stored or queried.
- *Insider Attacks* – launched by the adversary who has compromised one or more nodes in the network. Based on the secrets captured from compromised nodes and by fully controlling the behaviors of compromised nodes, the adversary attempts to discover or infer the types of data generated, stored or queried. Insider attacks includes:
  - *isolated insider attacks* - launched by the adversary compromising one node, and
  - *colluded insider attacks* - attacks launched by the adversary compromising more than one node.

An attacker can also perform a denial of service (DoS) attack such that the storage nodes fail to receive the data report or query successfully. Even more, the attacker can fabricate data, thereby the storage nodes may not be able to process the query. However, such DoS attacks and report integrity violations would fail to compromise the privacy of the data type and query. Other attacks include signal jamming, data dropping, false data injection, and false response. How to defend against these attacks are beyond the scope of this paper. Our future work involves adjusting afore-mentioned attacks.

III. The Proposed ElliPS Scheme

A. Prelimiaries

In order to preserve the privacy of the data types associated with the sensor data, the data types should be transformed from plain-text to a cipher-text. The transformation should be performed before the data report is sent out from any sensor node. The transformation should hold certain properties: (i) using the transformed data type, the storage node should be

able to execute and answer queries from base station effectively and efficiently; (ii) by executing the query, the storage node gains no knowledge about the data type queried for (since the storage node can be compromised); and (iii) colluding with sensor nodes, storage node learns no secret.

1) Preliminary Approaches: An approach tending to achieve the aforementioned goals can be described as follows. A mapping function $p(.)$ (e.g., one-way hashing, secret or public key-based encryption, etc.) is shared among all sensor nodes and the base station, but not the storage nodes. Before a sensor node sends out a data report, the data type, say t, is transformed to $p(t)$. When the base station wants to query data with type t, it sends $p(t)$ instead of t to the storage nodes. This approach is not resilient to the compromise of even a single sensor node, because all sensor nodes share the same mapping function.

Alternatively, every sensor node u may have a node-specific mapping function $p_u(.)$ shared with the base station. This approach has the following problems: (i) the query size becomes high because the query intended for type t should specify all node-specific transformed values of t (i.e., $p_u(t)$) for all u; (ii) query privacy cannot be preserved if a storage node and a sensor node are compromised. In this case, the compromised sensor node u can share its mapping function $p_u(.)$ with the compromised storage node. Hence, for any incoming query that contains $p_u(t)$ for certain data type t, the adversary can immediately infer that the query is intended for data of type t.

2) Key Ideas of Our Solution: In order to address the problems of the preliminary approaches and thus truely accomplished the aforementioned design goals, we propose our ElliPS Scheme based on the following key ideas:

- First, to be resilient to sensor node compromise, each sensor node uses a node-specific function to transform the types of its data before sending the data to the storage node; i.e., the type (t) of each piece of data is converted to $t' = p_u(t)$, based on both the original type (t) and the ID (u) of its source sensor node.
- Meanwhile, to reduce the query size and to prevent the collusion between sensor nodes and storage nodes, the transformed data type (t') needs to go through a series of intermediate nodes, which we call *anonymizer nodes*, to further transform t' to $t'' = \Lambda(t)$ by removing the effect of source node ID.

Note that, with the above two rounds of transformations, first, the connection between t'' and t is hidden from storage nodes and all sensor nodes (including the source node of the data), and hence protects the privacy of data types in the presence of sensor nodes and/or storage node compromise; second, the query to any data type (t) only needs to specify the final transformation result (t''), which keeps the query size small.

3) Notations: Following is a list of notations used in presenting the scheme:

- q, l: parameters of any verifiably random elliptic curve domain septuple $T = (l; a; b; G; q; h)$ [13].
- $\mathbb{F}_q$: a prime finite field.
- ζ: a cyclic subgroup of $\mathbb{E}(\mathbb{F}_q)$.
- α: a point from ζ.
- $\langle u, t_u, d_u \rangle$: a 3-tuple data report generated by a sensor node, where u is the ID of the sensor node, d_u is the data content, and t_u is the type of d_u.
- $f_{A_i}(x)$: a univariate polynomial, in which the degree of x is λ_f (λ_f is a system parameter). It is used by anonymizer A_i to transform the sensor node ID and data type in the report.
- $h_{A_i}(x)$: a univariate function. It is a one-way hash unique function for each A_i. It is used by anonymizer A_i to transform the sensor ID in the report.
- $f_v(x), h_v(x)$: functions having similar properties as of $f_{A_i}(x), h_{A_i}(x)$ respectively. It is used by the storage node v for similar functionalities.
- $p(x)$: a univariate secret polynomial, in which the degrees of x is λ_p (λ_p is a system parameter). This polynomial is used by network controller for the construction of l-unique transformation function $p_u(x)$ at each sensor node u to anonymize the data type in its report.
- $\Lambda(x)$: a secret univariate polynomial constructed by the point multiplying α with $p(x)$.
- $P_{u,v}$: a path from sensor u to storage node v which includes m ($m > 1$) anonymizers: $A_1, A_2, \cdots, A_m$. The paths of any two sensors u and w, that is $P_{u,v}$ and $P_{w,v}$ can be same or different.

B. Overview of the ElliPS Scheme

Using an example shown in Fig. 1, we now provide a non-strict overview of the proposed scheme, called ElliPS (Elliptic curve based Privacy Scheme).

With the proposed scheme, the system has the following entities: ordinary sensor nodes (data sources), a sequence of anonymizers, a storage node, and the base station. Each sensor node u (u is the ID of this sensor) is preloaded with a key K_u shared exclusively with the base station and a unique polynomial share $p_u(x)$ derived from $p(x)$; $p(x)$ is a secret polynomial kept at the base station. A nice property of $p_u(x)$ is that it is impossible to derive $p(x)$ from $p_u(x)$. Each anonymizer A_i is preloaded with a key K_{A_i} shared exclusively with the based station, as well as unique polynomials $h_{A_i}(x)$ and $f_{A_i}(x)$ which are used to anonymize sensor node IDs and data types, respectively. Storage node v is preloaded with a key K_v shared exclusively with the based station, as well as unique polynomials $h_v(x)$ and $f_v(x)$ which are used to anonymize sensor node IDs and data types, respectively.

As shown in Fig. 1, sensor node u generates data with content d_u and type t_u. To protect the privacy of the data (i.e., both the type and the content) before sending the data to the storage node, u transforms the data using its unique secrets (i.e., $p_u(x)$ and K_u). The transformation results in $\left\langle u^{(0)}, t_u^{(0)}, d_u^{(0)} \right\rangle$, where $u^{(0)} \equiv u$, $t_u^{(0)} = p_u(t_u)$, and $d_u^{(0)} = \{d_u\}_{K_u}$.

The transformed type and content can be reverse-engineered by the source node; that is, given $\left\langle u^{(0)}, t_u^{(0)}, d_u^{(0)} \right\rangle$, node u can easily recover the data type t_u and the content d_u. This poses a serious security breach. For example, if node u is compromised and the transformed data item $\left\langle u^{(0)}, t_u^{(0)}, d_u^{(0)} \right\rangle$ is among the set of data satisfying a certain query, the adversary can immediately infer the type of data being queried to be t_1. In addition, the above transformation does not conceal the source node ID.

To remedy the problem, our scheme requires that a data item transformed by its source node must also go through a sequence of (at least one) anonymizers before reaching the storage node to maintain untraceability. Particularly in the example shown in Fig. 1, data item $\left\langle u^{(0)}, t_u^{(0)}, d_u^{(0)} \right\rangle$

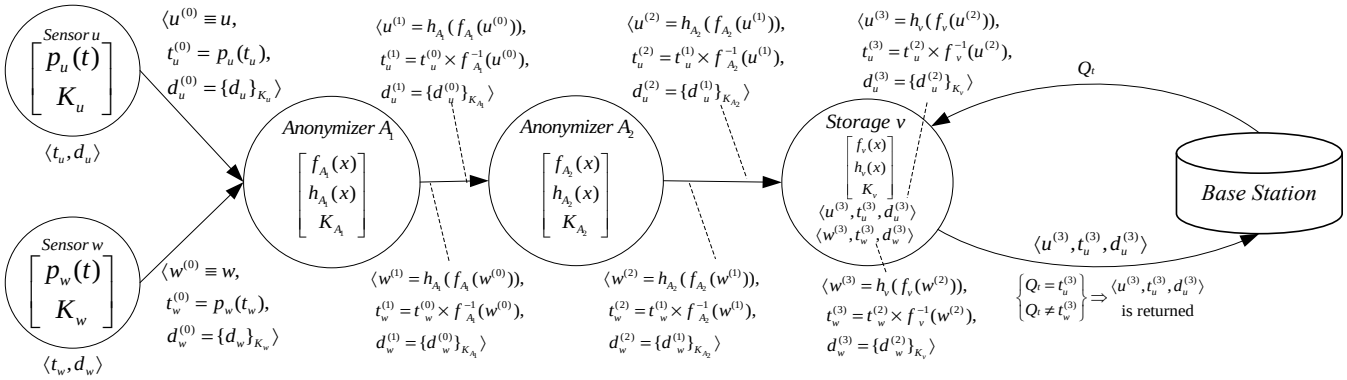

Fig. 1. An example to illustrate the proposed ElliPS scheme.

is transferred to anonymizer A_1, and is transformed by A_1 to $\left\langle u^{(1)} = h_{A_1}(f_{A_1}(u^{(0)})),\ t_u^{(1)} = t_u^{(0)} \times \frac{1}{f_{A_1}(u^{(0)})},\ d_u^{(1)} = \{d_u^{(0)}\}_{K_{A_1}} \right\rangle$. The data is then transferred to the next anonymizer node to get further transformation. This procedure continues until the last anonymizer on the sequence completes its transformation and sends the transformed data to the storage node, where the storage node performs similar transformation. Through these transformations, every sensor node cannot even recognize its own data item that are stored in anonymized format at the storage node; thus the connection between a data item and its source is broken. After passing m anonymizers, the anonymized data report stored at the storage node v takes the form $\left\langle u^{(m+1)} = h_v(f_v(u^{(m)})), t_u^{(m+1)} = t_u^{(m)} \times \frac{1}{f_v(u^{(m)})},\ d_u^{(m+1)} = \{d_u^{(m)}\}_{K_v} \right\rangle$. The property of the final transformed data type is given by: $t_u^{(m+1)} \equiv \Lambda(t_u)$ (we will elaborate this property in detailed description later in this section). In other words, $p_u(t_u)$ will be converted into $\Lambda(t_u)$ collaboratively by the anonymizers and the storage node.

The remaining issue is how type-based queries can be supported. Continuing with the above example, let us assume that the base station wants to query data of type t. The base station sends out a query, represented as $Q_t = \Lambda(t)$, to the storage node. Therefore, any data report of which $t_u^{(m+1)} == Q_t$, has the same data type (i.e, $t_u = t$). Hence, the storage node can return the queried data without even knowing the actual type of data being queried.

The scheme is designed based on function compounding techniques while constructing $p_u(x)$ and collaborative mix when the anonymizers are used to anonymize the data reports, where functions are defined over finite field denoted as $\mathbb{F}_q$. The function construction is based on elliptic curve group. The anonymizers and storage node collaboratively convert the transformed data type of form $p_u(t)$ (i.e, $t_u^{(0)}$, from any sensor node u) into the form $\Lambda(t)$ (i.e, $t_u^{(m+1)}$) when the data report reaches the storage node v. Independent of the data source, the data type t always takes the form $\Lambda(t)$ at the storage node. This allows an efficient query processing for the storage node, which include just mere value comparison (i.e, $t_u^{(m+1)} == Q_t$). By introducing the anonymizer the entire data report gets transformed, ensuring even the data source cannot recognize its own data report. Before presenting the details, we first list few notations and system initialization.

C. Detailed Description of the ElliPS Scheme

1) System Initialization: Based on the system requirements, the following steps are performed to bootstrap the system:

- The network controller preloads each node u (anonymizer, sensor or storage) in the network with a pair-wise key K_u shared with the base station. This allows all the nodes to communicate securely with the base station.
- *At Anonymizer Node (A_i):* The network controller randomly picks a unique polynomial $f_{A_i}(x)$, and a unique one-way hash function $h_{A_i}(x)$ for each anonymizer A_i.
- *At Storage Node (v):* The network controller randomly picks a unique polynomial $f_v(x)$, and a unique one-way hash function $h_v(x)$ for each storage node v.
- The network controller randomly picks a point α over ζ; it arbitrarily constructs a secret polynomial $p(x)$ over $\mathbb{F}_q$.
- *At Sensor Node (u):* The network controller arbitrarily selects a random path $P_{u,v}$ for sensor u, where v is a storage node for u. Let $\{A_1, A_2, \cdots, A_m\}$ denote the sequence of anonymizers along $P_{u,v}$. The network controller constructs a unique transformation function $p_u(x)$ for each sensor u as: $p_u(x) = \alpha_u \times p(x)$, where,

$$\alpha_u = \alpha \times \gamma_u, \quad \gamma_u = \prod_{i=1}^{m+1} \varrho_u^{(i)}, \quad \varrho_u^{(1)} = f_{A_1}(u),$$
$$\varrho_u^{(i)} = f_{A_i}(h_{A_i}(\varrho_u^{(i-1)})), \quad i = 2, ..., m+1.$$

Here, '$\times$' represents ECC point multiplication, both '$\cdot$' and '$\prod$' represent modular scalar multiplication. $p_u(x)$ is preloaded to u along with the ID of the first anonymizer along the path $P_{u,v}$, i.e, A_1. Subsequently, each anonymizer along the path $P_{u,v}$ is preloaded with the next anonymizer ID, except for the last anonymizer A_m which is preloaded with the ID of the storage node v. The network controller calculates the following list, $u^{(i)} = h_{A_i}(\varrho_u^{(i)}), \quad i = 1, ..., m$. Then for each path $P_{u,v}$, each anonymizer A_i is preloaded with the pair $\left\langle u^{(i)}, A_{i+1} \right\rangle$, where A_{i+1} is the next-hop anonymizer for a data report with $u^{(i)}$ as the ID element, except that anonymizer A_m holds $\left\langle u^{(m)}, v \right\rangle$. Similarly, along each path from every sensor node to the storage node, all anonymizers hold such pairs as routing information.

- *At the Base Station:* The network controller preloads the base station with all the secrets, including functions, path information and pair-wise keys. The network controller calculates ID mapping pairs for each path $P_{u,v}$ as follows.

It calculates $u^{(m+1)} = h_v(\varrho_u^{(m+1)})$ and stores to the base station a list of pairs $\langle P_{u,v}, u^{(m+1)} \rangle$ for each sensor u in the network. We will elaborate in the later section, how these ID mapping pairs help the base station to decrypt the encrypted data response from the storage nodes.

2) Sensor Node Behavior: Suppose sensor node u generates data with content d_u and type t_u; even though the data d_u has originated from sensor u, the data type t_u is only associated with the data content and not with the sensor u. The subscript u is used to show how the transformation works for data report from each sensor u. The sensor node u performs the following steps to generate the corresponding data report:

- Using the pair-wise key K_u shared with the base station, the sensor node u encrypts the data content d_u. The encrypted data content is represented as $d_u^{(0)} = \{d_u\}_{K_u}$.
- Using the preloaded transformation function $p_u(x)$, the sensor node u transforms t_u to $t_u^{(0)}$: $t_u^{(0)} = p_u(t_u)$ which in turn equals to

$$\begin{aligned} t_u^{(0)} = \alpha_u \times p(t_u) &= \left\{ \alpha \times \prod_{i=1}^{m+1} \varrho_u^{(i)} \right\} \times p(t_u) \\ &\equiv \alpha \times \left\{ p(t_u) \cdot \prod_{i=1}^{m+1} \varrho_u^{(i)} \right\}, \end{aligned} \tag{1}$$

  where m represents number of anonymizers in along the path from u to the storage node v, i.e., $P_{u,v}$.
- After the transformation, the 3-tuple report takes the form of $\left\langle u^{(0)} \equiv u, t_u^{(0)}, d_u^{(0)} \right\rangle$, which is then forwarded to the first anonymizer A_1 along $P_{u,v}$. Recall that the information about A_1 was preloaded to sensor node u during system initialization.

Our approach for transformation of data type and encryption of data content is based on the following idea. Data content is encrypted with the pair-wise key (shared between the sensor node and the base station) to protect the confidentiality. However, an adversary may still be able to infer some information about the encrypted data if the data type is not protected. Henceforth, in order to preserve the data type privacy, our scheme transforms the data type using the preloaded ECC-based function $p_u(x)$. Since $p_u(x)$ is exclusively shared between the sensor node and the base station, it is very difficult for the adversary to deduce the data type after such transformation (due to the hardness of the well-known discrete logarithm problem; detailed in Section IV). Since $p_u(x)$ at different sensor nodes are rooted from the same polynomial $p(x)$, the normal procedure of type-based queries at the storage is preserved, which will be shown in later in this section.

3) Anonymizer Node Behavior: Our scheme mandates that each data report goes through a sequence of anonymizers before reaching the storage node for additional transformations, in order to break the connection between a data report and its source. This ensures the property of data report untraceability. Specifically, when an anonymizer $A_i (i = 1, \cdots, n)$ receives a report $\left\langle u^{(i-1)}, t_u^{(i-1)}, d_u^{(i-1)} \right\rangle$, it performs the following steps:

- Using the pair-wise key K_{A_i} shared between anonymizer A_i and the base station, the encrypted data content in the 3-tuple report is further encrypted (we call this super encryption, which means encryption over encrypted data content). The encrypted version of the data content takes the form of $d_u^{(i)} = \{d_u^{(i-1)}\}_{K_{A_i}}$.
- Using the preloaded polynomial $f_{A_i}(x)$, the element $t_u^{(i-1)}$ in the 3-tuple report is transformed to $t_u^{(i)}$ as follows: $t_u^{(i)} = t_u^{(i-1)} \times \beta_i$, where $\beta_i = \frac{1}{f_{A_i}(u^{(i-1)})}$. We can also say, $\varrho_u^{(i)} \equiv \beta_i^{-1}$, where $^{(-1)}$ represents modular inverse (Proof of the existence of modular inverse is included in the technical report [14] due to space limitations. See Theorem A.3 in [14]).
- Using the preloaded function $h_{A_i}(x)$, the element $u^{(i-1)}$ in the 3-tuple report is transformed to $u^{(i)}$ as follows: $u^{(i)} = h_{A_i}(f_{A_i}(u^{(i-1)}))$. Recall that $u^{(0)} \equiv u$.
- The transformed 3-tuple report takes the form of $\left\langle u^{(i)}, t_u^{(i)}, d_u^{(i)} \right\rangle$, which is then forwarded to the next anonymizer along the path $P_{u,v}$. Recall that the ID of the next anonymizer along $P_{u,v}$ was preloaded to each anonymizer during system initialization.

4) Storage Node Behavior: The storage node v upon receiving the data report $\left\langle u^{(m)}, t_u^{(m)}, d_u^{(m)} \right\rangle$, it performs the following steps similar to anonymizer node:

- Using the pair-wise key K_v shared between storage node v and the base station, the encrypted data content in the 3-tuple report is further encrypted. The encrypted version of the data content takes the form of $d_u^{(m+1)} = \{d_u^{(m)}\}_{K_v}$.
- Using the preloaded polynomial $f_v(x)$, the element $t_u^{(m)}$ in the 3-tuple report is transformed to $t_u^{(m+1)}$ as follows: $t_u^{(m+1)} = t_u^{(m)} \times \beta_{m+1}$, where $\beta_{m+1} = \frac{1}{f_v(u^{(m)})}$. We can also say, $\varrho_u^{(m+1)} \equiv \beta_{m+1}^{-1}$.
- Using the preloaded function $h_v(x)$, the element $u^{(m)}$ in the 3-tuple report is transformed to $u^{(m+1)}$ as follows: $u^{(m+1)} = h_v(f_v(u^{(m)}))$.
- The final transformed 3-tuple report takes the form of $\left\langle u^{(m+1)}, t_u^{(m+1)}, d_u^{(m+1)} \right\rangle$, which is then stored for query processing.

The final transformed data type $t_u^{(m+1)}$ takes the following form independent of the source node where the data report has been generated:

$$t_u^{(m+1)} = \alpha \times p(t_u) = \Lambda(t_u) \tag{2}$$

where t_u is the type of data content d_u. Even though the data d_u has originated from sensor u, the data type t_u is only associated with the data content and not with the sensor u. In other words, if there exists $t_w^{(m+1)}$ whose $t_w = t_u$, then,

$$t_w^{(m+1)} = \alpha \times p(t_w) = \alpha \times p(t_u) = t_u^{(m+1)}. \tag{3}$$

This is because, collectively over each anonymizer and storage node, the transformation involves conversion of α_u to α. The theory behind this is explained below. We know,

$$\begin{aligned} t_u^{(0)} = p_u(t_u) = \alpha_u \times p(t_u) &= \left\{ \alpha \times \prod_{i=1}^{m+1} \varrho_u^{(i)} \right\} \times p(t_u) \\ &\equiv \alpha \times \left\{ p(t_u) \cdot \prod_{i=1}^{m+1} \varrho_u^{(i)} \right\}. \end{aligned} \tag{4}$$

Therefore, we can derive the following,

$$\begin{aligned} t_u^{(m+1)} &= t_u^{(m)} \times \beta_{m+1} \equiv t_u^{(0)} \times \left\{ \prod_{i=1}^{m+1} \beta_i \right\} \\ &\equiv \left\{ \alpha \times \left\{ p(t_u) \cdot \prod_{i=1}^{m+1} \varrho_u^{(i)} \right\} \right\} \times \left\{ \prod_{i=1}^{m+1} \frac{1}{\varrho_u^{(i)}} \right\} \\ &\equiv \alpha \times p(t_u) \equiv \Lambda(t_u). \end{aligned} \tag{5}$$

5) Base Station Behavior: Suppose the base station wants to query data of type t from the storage nodes. Clearly, if t is sent in plain text, the privacy of the query and the privacy of the data type satisfying the query are not preserved. Also as described earlier, the data types in 3-tuple data reports received by the storage nodes have already been transformed. Therefore, t should also be sent in the transformed format to preserve the query privacy as well as to facilitate the storage node to find matching data. In order for the base station to query data type t, the query is constructed as follows: $Q_t = \Lambda(t_u) = \alpha \times p(t)$. Then the base station forwards Q_t as query to storage node v.

6) Query Processing at the Storage Node: The storage node v is responsible for storing 3-tuple data reports from sensor nodes. It also processes each query Q_t from the base station. Upon receiving the query Q_t, for each stored data report $\left\langle u^{(m+1)}, t_u^{(m+1)}, d_u^{(m+1)} \right\rangle$, it performs the following verification. If $Q_t == t_u^{(m+1)}$, which means that the type of this data report is the same as the type queried by the base station, i.e, $t = t_u$ (according to the property of Q_t and $t_u^{(m+1)}$ as explained earlier). Therefore, all such matching reports of form $\left\langle u^{(m+1)}, t_u^{(m+1)}, d_u^{(m+1)} \right\rangle$ are returned to the base station.

7) Data Decryption at the Base Station: Upon receiving the reports of form $\left\langle u^{(m+1)}, t_u^{(m+1)}, d_u^{(m+1)} \right\rangle$, the data $d_u^{(m+1)}$ needs to be decrypted. The transformed ID $u^{(m+1)}$, preserves the order of transformation and encryption. In other words, preserves the path $P_{u,v}$ in which data report had been through. Knowing the $P_{u,v}$, the base station can infer the list of nodes (sensor, anonymizers, and storage) which encrypted the data over and over again.

The base station holding the list of all ID mapping pairs $\left\langle P_{u,v}, u^{(m+1)} \right\rangle$, can find the associated path $P_{u,v}$. Thereby, base station holding the pair-wise keys of all the sensors, anonymizers, and storage node in the path $P_{u,v}$ can decrypt the encrypted data eventually.

IV. Security Analysis and Discussion

In this section, we analyze the security properties of the proposed privacy solution. Following the attack models discussed earlier, our analysis will cover outsider attacks, and insider attacks, respectively.

A. Resilience to Outsider Attacks

An outsider is an adversary which is not a part of the network, trying to capture or infer some information about the data and query. The goal is to violate the privacy preserved in the system. The capability of such adversary is limited to eavesdrop the communication channel between sensors, anonymizers, storage nodes and the base station.

Having overheard a 3-tuple data report sent via open channel, the adversary may attempt to decrypt the data. Without compromising any nodes, the adversary can guess the keys. Since each bit of a key can be 1 or 0 with even chance, the probability for correctly guessing a key in one attempt is $\frac{1}{2^s}$ where s is the length of a key, and the computation complexity for finding out the key is $\Omega(2^s)$. Therefore, as long as s is large enough (e.g., 80 bits), the time complexity is prohibitively high.

Having captured a 3-tuple data report, the adversary may attempt to extract the data type of the report to infer some information or some secrets. Recall that, the data type has been transformed by the sensor node and/or some anonymizers. Specifically, if the report is captured on the link between a sensor node and an anonymizer, the data type has been transformed by the sensor node and has the form of $t_u^{(0)} = \alpha \times p(t_u)$ where u is the ID of the sensor node generating the report; if the report is captured on the link between two anonymizers, the data type takes the form of $\left\langle t_u^{(\omega)} = \alpha \times \{p(t_u) \cdot \prod_{i=\omega}^{m} \varrho_u^{(i+1)}\} \right\rangle (\omega = 1, 2 \cdots, m)$ where ω are the number of anonymizers that have transformed the report in the path $P_{u,v}$.

The interest of the adversary is to infer t_u from one or more captured $t_u^{(0)}$ or $t_u^{(\omega)}$. In either of the above two cases, without knowing α, the exact functions $p(x)$, and $\varrho^{(i+1)}(x)$(for $i = \omega, \cdots, m$), the adversary cannot find any relation from $t_u^{(0)}$ or $t_u^{(\omega)}$ to t_u, and hence has to guess t_u directly or the functions used for transformations, both needing brute force. Due to the hardness of the well-known elliptic curve discrete logarithmic problem (ECDLP), it is computationally infeasible to find α, $p(x)$ or $\varrho^{(j+1)}(x)$ from the shares($t_u^{(\omega)}$), for all $j = 1, 2, \cdots, m$.

Additionally, one cannot find the association between $\left\langle u^{(i)}, t_u^{(i)}, d_u^{(i)} \right\rangle$ and $\left\langle u^{(i+1)}, t_u^{(i+1)}, d_u^{(i+1)} \right\rangle$, or find the secrets used for transformation. This is because, a) the transformation of $u^{(i)}$ to $u^{(i+1)}$ involves function compounding of anonymizer specific secret functions, $f_{A_{i+1}}(x)$ and $h_{A_{i+1}}(x)$, known only to A_{i+1}; b) hardness of ECDLP in data type transformation; and c) encryption key used by each node is shared only with the base station. Since $h_{A_{i+1}}(x)$ is a one way function, λ_f plays no role in security or privacy level of the scheme if $\lambda \geq 2$ (see Theorem A.8 in [14]). Therefore, the scheme ensures untraceability of the report.

However, using extensive traffic pattern analysis, the adversary can physically map $u^{(i)}$ and $u^{(i+1)}$, in a specific path $P_{u,v}$; it allows the adversary to traceback the reports to the source node. The attack can be easily handled by having multiple paths and multiple transformation functions at each sensor node. Our scheme can handle such attack easily by introducing delayed reporting, report reordering and report mixing techniques at the anonymizers. Thereby, introducing anonymity among the reports being processed at the anonymizers [15], [16], it becomes hard for the adversary to find a unique mapping between $u^{(i)}$ and $u^{(i+1)}$.

Adversary monitoring the query and response channel, stores a list of transformed queries (Q_k $(k = 1, 2, \cdots)$) and data reports ($\left\langle u^{(i)}, t_j^{(i)}, d_j^{(i)} \right\rangle (i, j = 1, 2, \cdots)$); the query and the data type in the response takes the form $\Lambda(k)$. Due to the hardness of ECDLP, and the transformation function ($\Lambda(x)$) being kept secret at the base station, the privacy of data type remains intact.

B. Resilience to Insider Attacks

An adversary who has compromised sensor nodes, anonymizers and/or storage nodes may launch various attacks to compromise the privacy of data type and queries. Isolated insider attacks has limited impact over our scheme, because: (i) from a single compromised sensor node, the adversary cannot break the privacy of the data type generated by other innocent sensor nodes, since each sensor carry node specific transformation function and key; (ii) from a single compromised anonymizer, the adversary cannot break the privacy of any data report, because all the reports have been transformed collectively with secrets unknown to the compromised node; and (iii) from a single compromised storage node, finding the data type t_u from $\Lambda(t_u)$ has the hardness of ECDLP. Therefore, the following analysis focuses on attacks based on collusion of multiple compromised nodes.

1) Resilience to Collusion of Sensor Nodes: Suppose the adversary has compromised multiple sensor nodes, and the IDs of these nodes are $u_1, u_2, \cdots, u_\omega$. Adversary learns list of data types $\{t_1, t_2, \cdots, t_\tau\}$, where τ is the number of compromised data types. It may combine secret polynomials $p_{u_j}(x)(j = 1, 2, \cdots, \omega)$ captured from all these nodes to attempt to derive system-wide secret polynomial $\Lambda(x)$ or $p_\vartheta(x)$ where ϑ is a non compromised node. Since, $p_\vartheta(x) = \alpha \times \{\gamma_\vartheta \cdot p(x)\}$, γ_ϑ was constructed collectively with the secrets functions of m anonymizers and a storage node, adversary cannot derive $p_\vartheta(x)$(see Theorems A.6, & A.8 in [14]). Similarly, it can be shown, $\Lambda(x)$ remains secret with the hardness of ECDLP.

For every captured $\left\langle \vartheta^{(0)} \equiv \vartheta, t_\vartheta^{(0)}, d_\vartheta^{(0)} \right\rangle$, the adversary cannot derive t_ϑ if $\vartheta \notin \{u_1, u_2, \cdots, u_\omega\}$. This is because, a) each sensor node ϑ has node-specific transformation function $p_\vartheta(x)$, and b) it is computationally infeasible to learn secrets of other non-compromised node. If $\vartheta \in \{u_1, u_2, \cdots, u_\omega\}$, adversary can derive t_ϑ with a time complexity of $O(\tau)$(as defined earlier, τ is the number of compromised data types).

2) Resilience to Collusion of Anonymizers: If the adversary has compromised multiple anonymizers, and the IDs of these nodes are $A_1, A_2, \cdots, A_\omega$. It may attempt to utilize the information, $\left\langle f_{A_j}(x), h_{A_j}(x) \right\rangle (j = 1, 2, \cdots, \omega)$, captured from these anonymizers to break the privacy of data type or query, or learn some secrets. The adversary cannot succeed because of the hardness of one way function compounding of anonymizer and storage specific secret functions, and due to the hardness of ECDLP in ECC-based construction of transformation function. Compromising all the anonymizers may favor adversary to traceback source ID of a data report. However, learning the source ID is not sufficient enough to break the privacy of data type or query. Since the effect of sensor ID in the transformed data type is removed only at the storage node, as long as storage node remain non-compromised privacy of data types or queries remain intact.

3) Resilience to Collusion of Sensors and Storage Nodes: If the adversary has compromised multiple sensor nodes and a storage node, it may attempt to utilize the information captured from these two categories of nodes to break the privacy of data type or query. The attacks cannot succeed because all data reports received by the storage nodes have been transformed by anonymizers between them, which disconnect the links between sensor nodes and the reports they have generated. This is because: we know $t_u^{(0)} = \alpha \times \{\gamma_u \cdot p(t_u)\}$, and $t_u^{(m+2)} = \alpha \times p(t_u)$; since α, $p(x)$ and γ_u are kept secret, we have the hardness of ECDLP to find those secrets from shares or relate the data types.

4) Resilience to Collusion of Sensors, Anonymizers, and/or Storage Nodes: Suppose the adversary has compromised multiple sensor nodes, anonymizers and storage nodes. If the compromised anonymizers cannot form a complete path between the compromised sensor and storage nodes, the links between transformed data reports and sensor nodes are still broken since the reports have been transformed by some anonymizers that are not compromised. Therefore, the privacy of data type and query is preserved. If some compromised anonymizers form a path between the compromised sensor and storage nodes, the types of queries will be exposed. The attack is described as follows. Suppose a compromised storage node receives query Q_t. Sensor node u has been compromised, and all the anonymizers along the path between itself and the storage node have been compromised as well; hence, sensor node u can recognize any data reports from itself even if the reports have been transformed. If there exists a transformed data report $\left\langle u^{(m+1)}, t_j^{(m+1)}, d_j^{(m+1)} \right\rangle$ such that $Q_t = t_j^{(m+1)}$, the adversary will conclude that query Q_t is for date type t_u. Our proposed solution, however, can protect the privacy of data type if at least one of the nodes (i.e., reporting sensor, anonymizers, and the storage node) in the data reporting path remain non-compromised.

Over the path from a sensor node to a storage node, the data type transformation involves conversion of $\alpha_u \times p(t_i)$ to $\alpha \times p(t_i)$ based on ECC and modular inverse. Upon existence of one non-compromised anonymizer A_j, the functions $f_{A_j}(x)$, and $h_{A_j}(x)$ will be kept secret. Similarly, upon existence of non-compromised storage v, the functions $f_v(x)$, and $h_v(x)$ will be kept secret. In either case, due to the hardness of ECDLP and one-way hash function, the extensive collusion attack will not benefit in solving α or $p(x)$ (see Theorem A.8 in [14]).

For decrypting the encrypted data, adversary has to compromise or collude with all the anonymizers, storage and sensor node which are involved in encrypting the data over the path. Collusion of partial set of nodes has no benefit to decrypt the data report went through the path $P_{u,v}$. Therefore, the complexity is as in outsider attack ($\Omega(2^s)$).

Remarks: Based on above analysis, as long as at least one storage node or one anonymizers (among m) remains non-compromised (considering sensors are easily compromisable), our scheme preserves the privacy of data type, and query.

V. Performance Analysis and Evaluation

In this section, we present the implementation results of ElliPS on top of the Telos Mote/TinyOS platform. We measure the computation, communication and storage overheads for both regular sensor nodes, anonymizers and storage nodes.

A. Experimental Setup

Our experimental system consists of four types of nodes: regular sensor nodes, anonymizer nodes, a storage node and a base station. Based on recommended parameters defined in secp161r1 [13], the system parameter q is chosen to be the order of G from the elliptic curve domain septuple $T = (l; a; b; G; q; h)$. Based on T, the size of operands is set to 161

bits. Based on the security analysis, we set λ_p to 2 and vary λ_f between 2 and 80. Our implementation code reuses the ECC implementation contributed by Wang *et al.* [17]. Results are measured and averaged over 100 experimental runs.

B. Experimental Results

1) Computation Overhead: Each sensor node consumes approximately 3.421 seconds of CPU cycle time for transforming a data type (where $\lambda_p = 2$, which is sufficiently secure as we discussed in Section IV). Table I shows the computation overhead for one data report transformation at an anonymizer or the storage node in terms of CPU running time. The anonymizer consumes approximately 2.659 seconds as the maximum and 1.664 seconds as the minimum for transforming one item, as λ_f varies between 80 and 2. Though the operation needs 1.664 to 2.659 seconds, it is not conducted often if the frequency of producing data report is not high. When the data report is forwarded through other intermediate sensor nodes (other than anonymizers), no computation is needed.

TABLE I
COMPUTATION OVERHEADS FOR THE ANONYMIZER AND THE STORAGE NODE (IN SECONDS) - TELOS MOTES

Degree (λ_f)	80	40	20	10	2
CPU Cycle Time (sec)	2.659	2.217	1.903	1.781	1.664

When a sensor node produces a data report, it needs to perform one encryption for protecting data content, and λ_p scalar point multiplications and λ_p scalar point additions for transforming one item, i.e., evaluating $p_u(x)$.

When an anonymizer or storage node receives a data report, it needs to perform: (i) λ_f big integer modular multiplications and additions for evaluating $f_{A_i}(x)$ or $f_v(x)$, (ii) one big integer modular inverse and one hash operation for evaluating $h_{A_i}(x)$ or $h_v(x)$, (iii) one scalar point multiplication for final transformation of data type, and (iv) one super encryption over the data of each data report. Values of λ_p and λ_f are set to achieve the required security level. Query processing at storage nodes involves comparison between data type and query, whose computation overhead is negligible.

The data report transformation at each anonymizer causes accumulated processing delay (equivalent to computation overhead) when the data reports are transmitted between anonymizers. Note that, the delay is much less when the reports are forwarded between regular sensor nodes. The accumulated delay in such path has a lower bound equivalent to $\Omega((m+1) * C_{comp})$, where m is the minimum number of anonymizer in the data reporting path (such path may include intermediate regular sensors, which just perform forwarding) and C_{comp} is the computation cost for each anonymizer (e.g. $2.659s$ max, as shown in Tab. I). Since our current data storage and query architecture is focused on non-realtime query applications, we consider such accumulated processing delay is an acceptable trade off for privacy. The overall computation overheads experienced by regular sensor nodes, anonymizers and storage nodes all fall under the reasonable range, which in turn assures the feasibility of the proposed solution in the current generation of sensor nodes such as Telos Motes.

2) Storage Overhead: The program and data are initially loaded into the EPROM, and then loaded into the RAM to execute. We develop a stand alone program for testing. We measure the ROM and RAM consumptions, and results are shown in Table II. Considering the sizes of RAM and ROM in Telos Motes, the space requirements of about 1.351 KB RAM and about 20.6 KB ROM are affordable for sensor nodes. Similarly, the space requirements of about 1.353 KB RAM and about 22.2 KB ROM are also affordable for anonymizers and storage nodes.

TABLE II
STORAGE OVERHEAD FOR THE SENSOR NODE, THE ANONYMIZER AND THE STORAGE NODE (IN BYTES) - TELOS MOTES

	Sensor Nodes	Anonymizer/Storage Node
ROM	20690	22266
RAM	1351	1353

3) Communication Overhead: Our ElliPS scheme does not affect hop-by-hop communication cost when the size of the data type is also 161 bits. Otherwise, the communication overhead will be increased by the difference between 161 and the size of the data type. In the case of 32-bit data type, the overhead is $161-32 = 129$ bits, and in case of 64-bit data type, the overhead is $161-64 = 97$ bits. Such overhead is affordable compared with the size of the data report being transmitted. For each data report traversing over the data reporting path, the overall communication cost has a lower bound of $\Omega((m+\mu+1) * C_{comm})$, where m is the minimum number of anonymizer in the data reporting path, μ is the number of intermediate forwarding sensors and C_{comm} is one hop-by-hop communication cost. Note that, transferring data from a reporting sensor to its first anonymizer, or from one anonymizer to another, may need to pass other regular sensors; in such cases, the intermediate nodes simply forward the data without any further processing.

Afore-mentioned computation, communication and storage overheads offer a higher level of privacy for data type and query, which is the tradeoff for the privacy level offered by our solution. Our current implementation is limited to Telos Motes for ElliPS scheme; in future we planned to test the overheads in other sensor platforms like mica2, micaz, etc.

VI. DISCUSSION

A. Open Issues

Our proposed scheme performs well in preserving data type and query privacy against outsider attacks and insider attacks, regardless whether there is collusion among sensor nodes, anonymizers, and storage nodes. Still there is one possible way for an insider to attack our scheme. At the storage node, there exists an one-to-one mapping between a data type (t_u) and its transformed data type ($\Lambda(t_u)$). Presence of limited number of data types allows the adversary to learn this fixed mapping; however, including the temporal attribute for each data type increases the complexity for finding such mapping. Even though the privacy of the data type remains preserved, grouping data reports which share common transformed data type is feasible. Eventually, a compromised storage node processing queries may learn the frequency of queried data types. Therefore, the query pattern may be revealed to the adversary. Further, sensor network is prone to node failures. Therefore, in a data reporting path, one or more anonymizers may fail; the data report traversing through such a path will fail to reach the storage node.

B. A Possible Solution

In this section, we discuss a possible solution to handle the issues, namely, i) fault tolerance to anonymizer failure, and ii) query pattern privacy. We introduce multiple paths with multiple transformation functions to tolerate anonymizer failure. Therefore, the reporting sensors can pick one of the available paths to send the data reports. To preserve the privacy and freshness of the query pattern, we introduce batch query processing with added randomness in the query.

1) Multiple Paths: During the system initialization, the network controller randomly picks l unique points $(\alpha_1, \alpha_2, \cdots, \alpha_l)$ from ζ; and it finds l unique paths from each reporting sensor u to storage node v, where each path includes m ($m > 1$) different anonymizers: $A_1, A_2, \cdots, A_m$. For each path $P_{u,v}^{(j)}$, network controller preloads the sensor u with one unique transformation function: $p_{u,j}(x) = \alpha_{u,j} \times p(x)$, where $\alpha_u = \alpha_j \times \gamma_u$ for $j = 1, 2, \cdots, l$. Here l is a system parameter, and γ_u is constructed as described in Sec. III-C1.

For any two sensors u and w, the paths $P_{u,v}^{(j)}$ and $P_{w,v}^{(k)}$ could be the same. However, the transformation functions $p_{u,j}(x)$ and $p_{w,k}(x)$ are different (for all $j = 1, 2, \cdots, l$ and $k = 1, 2, \cdots, l$). Upon data report generation, the sensor randomly picks one of the l paths and transforms the data type using the associated transformation function. Then the sensor forwards the data report to the storage node through the chosen path. At the storage node, the transformed source ID in the data report is path specific, which helps the Base Station to decode the path of traversal for each data report when returned. However, the transformed data type at the storage node remains independent of chosen path, allowing efficient query dissemination and processing.

Neighborhood node monitoring techniques [18] can be used for reporting anonymizer failures and available active paths throughout the network. Reporting sensors can then pick one of the active paths for sending their data reports. Introducing l multiple paths provides a fault tolerance threshold of $(l-1)$ path failures; note that, a network controller can adjust l based on the anonymizer failure rate, when the network is deployed.

2) Batch Query Processing: Queries sent from the base station are processed in batches, where the user queries are buffered at the base station and forwarded to the storage nodes batch by batch. Each batch query contains a group of data types of users' interests. Privacy of the query pattern and the freshness of the batch query will be preserved when the base station (i) introduces random data types in each batch query, and (ii) randomly changes the size of the batch each time a batch query is forwarded to the storage nodes.

The batch query is constructed by using a Bloom filter. The network controller picks k independent hash functions and preloads them at each storage node and the base station. Suppose the base station wants to query data of types $Bat = \{t_1, t_2, \cdots, t_{\eta-\psi}, \cdots, t_\eta\}$ from storage nodes, where η is the size of the batch and ψ is the number of randomly introduced data types for querying. For each data type $t_i \in Bat$, base station calculates a list of transformed data type $Q_{t_i,j}$ as follows: $Q_{t_i,j} = \Lambda_j(t_i) = \alpha_j \times p(t_i)$, where $i = 0, 1, \cdots, \eta$; $j = 0, 1, \cdots, l$. Using the k hash functions, the base station constructs a Bloom filter Q_{Bat} of size M, to store elements $\{Q_{t_1,1}, \cdots, Q_{t_\eta,l}, Q_{t_2,1}, \cdots, Q_{t_2,l}, \cdots, Q_{t_\eta,1}, \cdots, Q_{t_\eta,l}\}$; then the base station forwards Q_{Bat} as a transformed batch query to the storage nodes.

Upon receiving Q_{Bat}, each storage node finds all the data reports belonging to Q_{Bat} using the preloaded hash functions. Query is processed as follows: k hash values are computed for the data type in each data report, and the data report is returned if the hash values indexed bits are all 1 in Q_{Bat}. Similar to Sec. III-C1, the base station can recover the data.

3) Future Work: Other topics that may need further investigation include anonymizer deployment, location privacy of anonymizers, and improvement in communication efficiency between storage nodes and the base station. We plan to address these topics in the future work.

VII. Conclusions

In this paper, we consider the in-network data storage and query architecture for sensor data management, and investigate the problem of protecting data type privacy and query privacy in the presence of sensor node compromise, anonymizer compromise or storage node compromise, or under collusive attacks by compromised sensor nodes, anonymizers and storage node together. We propose a solution called ElliPS, and verify its security properties and efficiency with extensive analytical and simulation studies.

Acknowledgments

The research reported in this paper was supported in part by the Information Infrastructure Institute (iCube) of Iowa State University and the National Science Foundation under Grants CNS 0716744, CNS 0831874 and CNS 0627354.

References

[1] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin, "Data-centric storage in sensornets," *in ACM SIGCOMM Computer Communication Review*, pp. 137 – 142, Jan 2003.

[2] Y. Diao, D. Ganesan, G. Mathur, and P Shenoy, "Rethinking Data management for Storage-centric Sensor Networks," *in Proc. Conference on Innovative Data Systems Research (CIDR)*, Jan 2007.

[3] W. Liu W. Lou and Y. Fang, "SPREAD: Enhancing Data Confidentiality in Mobile Ad Hoc Networks," *IEEE INFOCOM*, Mar 2004.

[4] J. Luo, P. Papadimitratos, and J.-P. Hubaux, "GossipCrypt: Wireless Sensor Network Data Confidentiality Against Parasitic Adversaries," Tech. Rep., 2007.

[5] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," *in Proc. IEEE Symposium on Security and Privacy (Oakland)*, May 2000.

[6] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," *in Proc. the 3rd annual conference on Applied Cryptography and Network Security (ACSN)*, Feb 2005.

[7] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," *in Proc. the 4th IACR Theory of Cryptography Conference (TCC)*, Feb 2007.

[8] E. Shi, J. Bethencourt, T-H. H. Chan, D. Song, and A. Perrig, "Multi-Dimensional Range Query over Encrypted Data," *in Proc. IEEE Symposium on Security and Privacy (Oakland)*, May 2007.

[9] B. Carbunar, Y. Yu, L. Shi, M. Pearce, and V. Vasudevan, "Query privacy in wireless sensor networks," *in Proc. IEEE SECON*, Jun 2007.

[10] B. Sheng and Q. Li, "Verifiable privacy-preserving range query in sensor networks," *in Proc. IEEE INFOCOM*, Apr 2008.

[11] D. L. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," *in Communications of the ACM*, Feb 1981.

[12] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," *in Proc. the 13th USENIX Security Symposium*, Aug 2004.

[13] Certicom Research, "SEC2: Recommended Elliptic Curve Domain Parameters," *in Std.'s for Efficient Cryptography version 1.0*, Sep 2000.

[14] N. Subramanian, K.Yang, W.Zhang, and D.Qiao, "*ElliPS*: Privacy Preserving Scheme for Sensor Data Storage and Query," *Technical Report, www.cs.iastate.edu/∼nvsubram/subra-ka-privacy08.pdf*, 2008.

[15] J. Deng, R. Han, and S. Mishra, "Countermeasures Against Traffic Analysis Attacks in Wireless Sensor Networks," *in Proc. IEEE SecureComm*, Sep 2005.

[16] J. Deng, R. Han, and S. Mishra, "Intrusion Tolerance and Anti-Traffic Analysis Strategies for WSN," *ICDSN*, Jun 2004.

[17] H. Wang and Q. Li, "Efficient implementation of public key cryptosystems on mote sensors (short paper)," in *International Conference on Information and Communications Security (ICICS)*, 2006.

[18] I. Khalil, S. Bagchi, and C. Nita-Rotaru., "DICAS: Detection, Diagnosis and Isolation of Control Attacks in Sensor Networks," *In IEEE Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm)*, Sep 2005.