Building Sensor-Based Big Data Cyber Infrastructures

Elisa Bertino[1], Surya Nepal[2], Rajiv Ranjan[3]

[1]Department of Computer Science, Purdue University, USA
[2]Data61 (Previously Digital Productivity Flagship), CSIRO, Australia
[3]School of Computing Science, Newcastle University, UK

Introduction

Novel applications in the domain of precision agriculture, biophysical sciences, and environmental sciences utilize sensors to collect a multitude of data including weather, plant pathogens, vehicle movement, airborne contaminants, energy distribution and water consumption. Recent advances in nanotechnologies will soon also make it possible to deploy nanosensors able to collect data at a very fine granular scale. The recent Internet of Things (IoT) trend will further push the deployment of sensors and embedded systems to a variety of application domains, including industrial manufacturing, networks of medical devices and personalized remote healthcare. These IoT sensors generate a large volume of data that needs to be processed and analysed to build a smart decision support system specific to each application type. While the emerging cloud and big data processing technologies provide the ability to store and efficiently process large scale data sets, there exists a significant research gap as regards to their inter-operation with the processes (e.g. deployment, provenance, security, and data processing) governing sensors

Figure 1 presents a conceptual architecture [1] for sensor-based big data cyber infrastructure. The layered architecture contains three layers. At the bottom layer, it has sensor networks that collect and sense a large set of data in real-time. The middleware layer includes the cloud-based big data processing technologies. Previous BlueSkies columns "processing distributed internet of things data in clouds" [2] and "streaming big data processing in datacentre clouds" [3] have already provided different aspects of these technologies at this layer. Finally, the top layer is the application layer. It includes a variety of applications ranging from precision agriculture to personalised health care [4][5]. The focus of this article is on the needs of tools and technologies at the middleware that can facilitate efficient management of and inter-operation with the Sensor Network layer, highlighted in different colours in Figure 1.

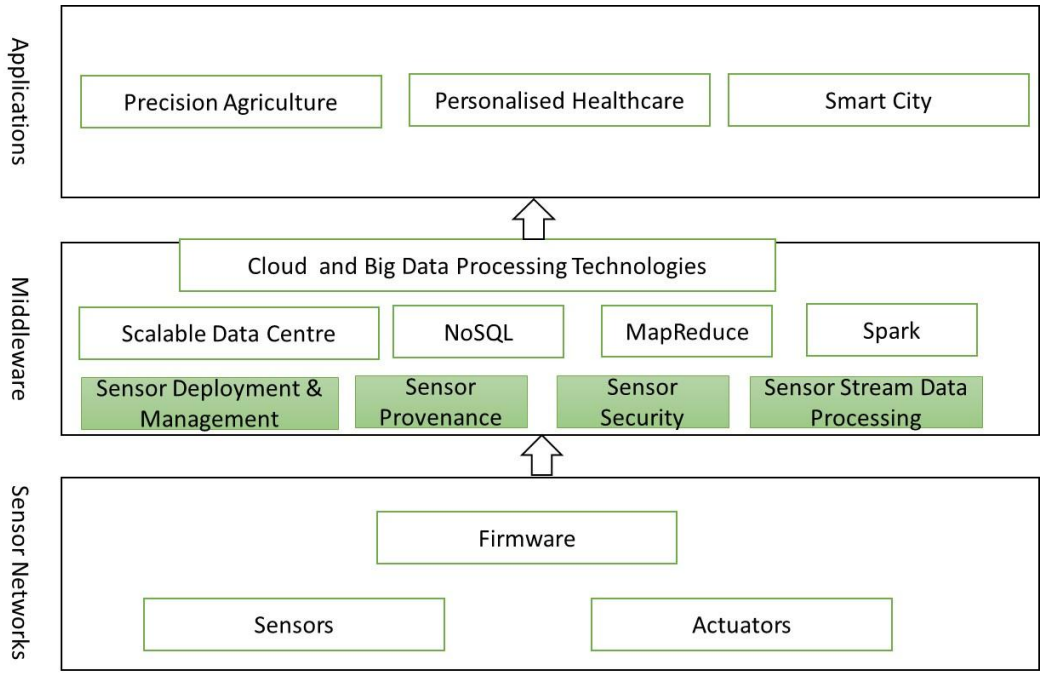

Figure 1: A conceptual architecture of sensor-based data intensive cyber infrastructures.

Challenges

Today cloud and big data technologies provide the ability to store and efficiently process large scale data sets in the middleware layer by offering a mix of software and hardware resources as discussed in previous instalment of Blue Skies [2][3]. However, the success of the applications at the top layer critically depends on the Integration and access to information collected by sensors at the bottom layer. Hence, there are number of challenges that need to be addressed in order to design and implement the next generation sensor-based big data cyber infrastructures. We next describe the key challenges

- Lack of tools to aid the design of sensor-based data collection applications. The design of applications that involves data acquisition from sensors requires making many configuration selection decisions, such as the type of sensors to be deployed and their battery life, and where to place the sensors and the respective gateways.. For example, in the case of sensors deployed in rivers, the location directly affects the collected data and resulting decision making ability (e.g., flooding prediction). On the other hand, in context of precision agriculture when conducting broad scale phenotypic measurements, unforeseen soil variations (e.g., fertility, texture, organic matter, pH, water status, compaction, etc.) increase the statistical variance of individual plant measurements. As a result, it is unclear what sensor placement density should be employed to accurately detect meaningful differences among biological treatments and for specific classes of sensors for precision agriculture. It seems that historical data concerning sensor placement performance from past experiments be leveraged for making above configuration selection decisions in context of new sensor-based data collection applications.
- Lack of tools to aid the deployment and monitoring of sensors. Continuous monitoring of sensors is essential in order to prevent data losses, and to dynamically adjust sensor performance during critical experimental events (e.g. fine resolution measurement of wind speed during a thunderstorm). In many cases, sensor data is not recoverable if the loss occurs due to inefficient caching and communication protocols. And if that specific data is critical to understanding the performance of experimental applications (e.g., crop yield within a field-based genetic test), the entire experiment can be lost for that year. For this reason, continuous monitoring and early detection and correction of failures are critical. Additionally the ability to observe accumulated results in real-time is necessary to ensure data integrity as it is collected.
- Heterogeneity in sensor capabilities and wireless link quality. Current sensors implement myriad of communication protocols. Further, it is well understood that wireless communication links are affected by weather and crop canopies. Moreover, batteries in sensors may not be easily replaceable, so all protocols must be energy-efficient. Finally, sensor clocks may drift, which can significantly impair duty-cycling and sleep/wake scheduling protocols.
- Lack of provenance and metadata tools for sensors. Provenance is essential for keeping track of which data was collected and/or aggregated for which sensors, including mobile sensors, such as unmanned aircraft systems (UAS). Metadata is also essential for conveying contextual information about sensor deployment and life-cycle. Conventional approaches require collection and transmission of large data volumes. Techniques are thus needed that reduce the size and enhance the efficiency of provenance and metadata collection, recording, and transmission.

Addressing the above gaps requires designing and developing novel cyber infrastructures to automate the efficient collection, normalization, curation, measurement and sharing of sensor data. In what follows we elaborate on key requirements of such infrastructures as well as on approaches and research directions to address some these requirements.

Sensor Deployment and Management

The first key requirement for the development of cyber infrastructure component is the availability of the techniques that can support the proper placement of sensors and their management and localization. Localization is an important issue in the sense that for many sensor-based applications data provided by sensor must be geo-referenced. Therefore, supporting tools are needed for automatic geo-referencing of sensors.

The second important requirement is the development of sensor placement techniques. Proximal sensing requires placement of the sensor at a close range or even in contact with the entity being monitored, for example with the soil being measured in the precision agriculture application domain. In such an application

domain, while it is sufficient for some applications to have a single site measurement, it is more typical to have a soil properties map of the entire field. Apart from soil monitoring, sensors can be used to monitor the growth of crops. In addition, different crops have different sensitivity to soil conditions and are planted at different densities. Therefore, effective sensor placement for precision agriculture application domains as well as for other domains require to consider: (1) domain specific knowledge, (2) sensor failure and attack probability, and (3) wireless link quality. Tools are thus needed to support optimal application-dependent sensor placement.

The third critical requirement is related to the optimal configuration of the duty-cycling, sleep/wake scheduling, and routing of sensors. The duty cycle must be controlled by the application. For example, in a precision agriculture application depending on the crop life cycle, it may be important to control the duty cycle of sensors: some sensor data may be important and others may not be during a particular season or month. For example, in Eastern Washington State, climate can cause low-temperature plant injury. On days when a frost or freeze event is forecasted, workers start monitoring air temperatures in early evening hours and begin to shut down soon after sunrise when temperatures rise above critical temperatures. During the following day, plants are checked for damage and equipment is made ready for the next event. For some weather systems, this cycle may last for several days. In such critical scenarios, sensor readings should be collected more frequently than in the case of normal weather to avoid potential plant injury. Hence, in the above scenario the cyber infrastructure needs to include a decision support system which has capability to dynamically and minimally adjust the push/pull rates for each sensor, according to the real-time data processing needs. A reactive programming approach can be useful in this regard to let programmers define a set of rules for each sensor for how frequently and on which conditions data should be reported to the base station.

The fourth requirement is the need of techniques assuring robust communication under poor link quality. Wireless link quality changes over time due to seasonal changes in foliage size/extent, variations in plant density, weather/wind/moisture, and seasonal change in crop canopy height and density. Data dissemination with such variable link quality leads to several challenges, such as reliability degradation, redundant transmissions, and increased contention among node transmissions. The resulting varying link correlation observed in real networks makes the problem harder. Multi-packet flooding protocols must be investigated that exploit the synergy among link correlation and network coding. One can take advantage of link correlation to eliminate the overhead of explicit control packets in networks with high correlation, and use network coding to pipeline transmission of multiple packets, attempting to utilize only a single timer per node for increased scalability.

Finally the last requirement is to be able to continuously monitor sensors to verify that they work as expected. Monitoring sensors is a complex task as there could be multiple factors that could lead to sensor malfunctions such as calibration errors,environmental conditions, attacks, decay of sensor energy, etc. Different monitoring techniques may thus have to be combined and deployed, ranging from simple techniques, such as profiling sensor baseline behaviour, to complex techniques, such as fine-grained diagnosis techniques for sensors [6].

Sensor Provenance

Provenance of sensor data is critical in many applications as it: (i) plays a key role in assessing and ensuring data trustworthiness; (ii) aids in preventing data losses; (iii) ensures the repeatability of scientific experiments and processes; and (iv) helps in preventing and investigating scientific frauds. Having said that provenance has to be fine-grained in order to be able to track which sensors acquired and/or transmitted specific data items. Further, location and time of the data acquisition is also critical as they are often the basis for data integration. Moreover, Ii is also crucial that provenance information includes all processing steps executed on data, such as data conversion steps, and data integration and aggregation steps. These steps may be executed on different subsystems within the middleware, including map-reduce systems, relational DBMS, stream processing systems, distributed in-memory data stores, and NoSQL databases. The hard research challenge is to capture such provenance operations across multiple, heterogeneous big data processing technologies and sub-systems. Finally, it is critical that provenance be secured to prevent provenance information from being tampered and/or lost. In this section, we first introduce key element of a suitable provenance model and then we discuss the issue of provenance transmission.

Provenance Model

One of the important step in the development of tools for managing provenance information is to develop an expressive provenance model which is capable of representing the provenance of data objects with various semantics and granularity. Such a model should be able to capture data provenance in a structured way as well

as to encapsulate the knowledge of both the application semantics and the system. At the same time it should support provenance interoperability. One such model has been recently developed by Sultana and Bertino [7] (see Figure 2); the developed model addresses various requirements, including expressive power, flexible level of provenance granularity, and security, and is compatible with existing provenance standards.

In the model by Sultana and Bertino [7], data creation or manipulation is performed by a sequence of *operation*s initiated by a *process*. A *process*, consisting of a sequence of operations, may be a service/activity in a workflow, a user application, or an OS-level (e.g. UNIX) process. An *operation* executes specific task(s) and causes manipulation of some system or user data. Thus, the operations not only generate/modify persistent data but also generate intermediate results or modify system configurations. *Communication* represents the interaction (e.g. data flow) between two processes or two operations in a process. Communication between two operations in a process means the completion of an operation following the start of another operation. When the preceding operation results in data, the communication may involve data transmission between the operations. The communication may also contain triggers, specific messages, etc. However, in most of the cases there might be no explicit message (i.e. communication record) exchange between two operations. Web service, user application, and UNIX process are examples of *process*es; statements within an executable, function, command line, etc. exemplify the *operation*s; while data flow, copy-paste, inter-process communication in UNIX, etc. represent the *communication* between operations or processes. An operation may take data as input and output some data. Each data object is associated with a *lineage* record which specifies the immediate data objects that have been used to generate this data. *Lineage* is particularly helpful for producing the data dependency graph of a data object. Processes, operations, and communications are operated by *actors* that can be human users, workflow templates, etc. When data provenance is used to detect intrusion or system changes, the knowledge of a user role or the workflow template may be helpful. *Environment* refers to the operational state, parameters, system configurations that also affect the execution of an operation and thus output data. This additional provenance information is crucial for understanding the performance of the operation and the nature of the output. In addition, the model supports the specification of *granularity policies*, allowing the users to specify how detailed the provenance information is to be captured and stored. It also includes, as part of provenance, information on the access control policies under which data have been accessed, which is crucial to investigation of data breaches.

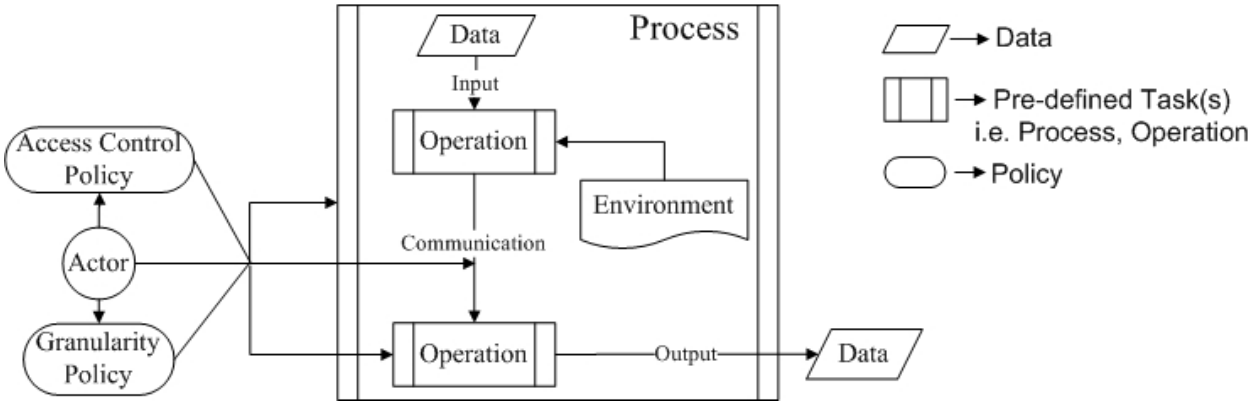

Figure 2. Provenance Model

A relevant open research direction concerning the use of such model within cyber infrastructures for sensor management is the definition of services to be associated with the provenance data. To effectively use the provenance information it is critical that the model supports various types of provenance queries. Historical dependencies as well as subsequent usages of a data object should be tracked easily. If a data is processed across multiple big data processing technologies and subsystems (see Figure 1), an administrator might want to see a high level machine, system or domain view of the provenance graph. In addition, to find relevant information from large provenance graphs, one should be able to filter, group or summarize all/portions of provenance graphs and to generate tailored provenance views. Thus, the model should be able to distinguish the provenance generated from different systems and construct specialized views of provenance graphs. Examples of queries to be supported include:

- *Fundamental Queries on Entity Attributes*: These queries retrieve information about the fundamental entities of the provenance model. Examples of such queries are: find all the operations executed by a given process, and generate the sequence of processes/operations in a workflow. These queries can help in detecting anomalies by comparing the expected output of an operation in the recorded environment with the actual result. Users that have executed anomalous operations can be identified by finding out the actors that invoked the operations.
- *Queries on Invocations:* These queries retrieve the set of commands involved in the manipulation of a selected data object. Users can set various filters while retrieving the provenance, such as remove commands that occurred before or after a given point of time. These queries help users in reproducing a data object, detecting system changes or intrusions, and finding out the system configuration during process invocation, understanding system dependencies, etc.
- *Queries on Lineage:* The historical dependencies of a data object can be determined by traversing the provenance graph backward whereas data usage can be traced by forward traversal of the graph. A simple query can be of the following form: "*find the ancestor data objects to data d"*. More complex queries may refer to patterns within the derivation graph.
- *Provenance Views:* Since provenance grows fast, it might be convenient (often required) to compress or summarize the provenance graph for efficient querying and navigation. For example, instead of keeping track of how different processes and people modified a data set five years ago, we can replace the part of the provenance with the end result of the modification. A modularized provenance model, such as the one by Salmin and Bertino [7], would be able to support queries to generate any abstraction of a provenance graph. The *domain* attribute in the provenance records greatly helps in writing a quick and effective abstraction function for an intended purpose.

Provenance Transmission

In a multi-hop sensor network, data provenance allows the base station to trace the source and forwarding path of an individual data packet since its generation. However, tight storage, limited resources, energy and bandwidth constraints of the sensor nodes raise important challenges in the scenarios where an administrator want to undertake data provenance for each packet. Hence, a great deal of efforts has been devoted to design lightweight data provenance schemes [8][9], for wireless sensor networks. In these schemes, data provenance is represented as a directed graph, where each vertex represents the provenance record of a node in the data flow path and each edge between two nodes indicates the direction of data transmissions between the nodes. One major issue with these schemes is that the provenance size increases with the number of nodes in the data flow path. Due to the growing size of data provenance, per-packet provenance transmission in sensor networks, bandwidth and energy exhaustion may occur. Therefore, it is necessary to devise light-weight provenance solutions that provide the complete provenance of a data packet without introducing significant overhead.

A more recent approach has been proposed [10] whereby each node in the data flow path compresses and encodes its provenance record using arithmetic coding. Arithmetic coding [11] is a lossless data compression technique that assigns short code words to more probable events and longer code words to less probable events. Each code word is represented by a half-open subinterval of the half-open unit interval [0, 1). Enough bits are used for a code word to distinguish the corresponding subinterval from all other possible subintervals. The subinterval is refined as soon as the probabilities of individual events change. In such scheme, each sensor in a sensor network represents an event and the occurrence probability of the sensor is used for provenance encoding and decoding. With arithmetic encoding, such scheme achieves a provenance compression ratio that approaches to Shannon's theoretical entropy bound and thus outperforms, in most respects, the better-known Huffman coding based compression schemes. Unlike other approaches, the most important characteristic of such scheme is that the provenance size is not directly proportional to the number of hops. Thus, the scheme can save more energy and bandwidth of sensor networks compared to existing provenance schemes. However the deployment of such scheme in sensor networks requires addressing several additional issues including:

- How handle dynamic wireless sensor networks, such as sensor networks including mobile devices and drones?
- How to handles cases in which packets are routed in more than one node at a time?
- How to provide secure sensor location information as location information is critical for many sensor-based applications?

Sensor Security

Many scenarios of interest to sensor-based applications require assuring data integrity and confidentiality as well authentication among different sensors and other parties. For example, before a drone can exchange data with a sensor to determine the sensor location or to collect data from a sensor, the drone needs to authenticate with the sensor. This prevents malicious parties from tampering with sensor configurations which can in turn result in data losses. Also sensors deployed in a given context may belong to different projects or applications and therefore it is important that data collected by each sensor only be available to authorized parties. For example, a drone may be authorized to collect data only from a specific set of sensors on the ground and not from other sensors. Also sensor devices are vulnerable to malicious attacks such as impersonation, interception, capture or physical destruction, due to their unattended operative environments and lapses of connectivity in wireless communication [12]. To address security, encryption key management protocols need to be deployed. Current approaches, such as those based on symmetric key encryption, elliptic curve public key cryptography, identity-based public key cryptography, suffer from one or more of the following drawbacks: high communication and storage overhead, vulnerability to impersonation and key compromise attacks, certificate management overhead and computational overhead for pairing operations, inability to support mobility.

A novel pairing-free certificate-less public key cryptography based key management (CL-EKM) scheme for dynamic WSNs has been recently proposed that addresses the drawbacks of previous approaches [13]. In certificate-less public key cryptography (CL-PKC) [14], a party's full private key is a combination of a partial private key generated by a key generation center (KGC) and the party's own secret value. The special organization of the full private/public key pair removes the need for certificates and also resolves the key escrow problem by removing the responsibility for the party's full private key. The CL-EKM scheme supports the establishment of four types of keys for each sensor node: (i) a certificate-less public/private key pair for the sensor, (ii) an individual key shared with the base station, (iii) a set of pairwise keys, each for each sensor node in the same sensor cluster, (iv) a cluster key shared among all the neighbouring nodes. It is important to notice that when dealing with mobile sensors, the dynamic establishment of pairwise keys is essential to allow the mobile device (e.g. a drone) to securely communicate with a specific sensor on the ground (for example to issue reconfiguration instructions to the sensor or to collect data and provenance from the sensor). As different drones may be used at different times, the pre-installation of keys is not feasible and we thus need mechanisms to dynamically generate such keys as needed. Cluster keys are also critical, for example in case of collaborative activities by a group of drones and on-ground sensors. The CL-EKM scheme is thus able to address all these scenario and, as it does not require pairing operations, it is also highly efficient. Research is however needed to perform extensive experimental evaluation with different types of sensors and networks as well as to develop encryption schemes supporting data aggregation and processing at the sensor network level. Also in addition to securing the communication, techniques are required to secure the software running on the sensors, such as the adoption of suitable code randomization techniques [15], and to quickly respond to security incidents [16].

Sensor Streamed Data Processing Techniques

Two main strategies can be devised for processing the data acquired from sensors:

Data warehousing approach – it involves shipping raw sensor readings from the sensor network to a central repository stored in a cloud for subsequent analysis. In this case there is no in-network processing in the sensor network. This approach is most appropriate when all data needs to be kept and there is sufficient energy to transmit the data outside the sensor network and/or collection is possible (for example a drone is available to immediately collect the sensed data). This approach is also critical when the data requires computationally intensive processing. In recent times, there have been a number of open source data streaming tools that are made available to be used in the data warehousing approach such as Apache Samza, Esper, Spark Streaming, Apache Storm, and Apache S4. They can be classified in different categories based on the architecture and processing technique. For example, Apache Samza is a parallel distributed system, whereas Esper is a streaming system with a centralized architecture that runs on a single node and keeps everything (states, operators, and so on) in memory [3].

In sensor-network processing – it involves executing certain data operations in the sensor network itself and shipping outside the results of these operations. Typical examples include computing some simple aggregated functions, such as computing the maximum among a set of sensor readings, and performing outlier detection.

This approach is most appropriate when energy saving can be gained by locally processing the data, thus saving communication costs. Also this approach is critical when data quality needs to quickly analysed in order to assess whether the sensors are providing correct data and, if not, to locally implement some actions to restore data quality. It is clear that both approaches need to be support by cyberinfrastructures and used depending on the requirements of specific applications and data sensor projects. Approaches similar to cougar that provides in-network query processing [17] [18] need to be developed for processing streaming data.

Fog Computing is another emerging technology, though in the early stage of its development, that has potential to support in sensor-network processing [20]. It is a highly virtualised platform located at the edge of the network and provides compute, storage and network services between the end devices and the traditional cloud data centres. Its characteristics such as low latency, location awareness, support for large number of nodes, and wide spread geographical distribution make it suitable for streaming and real time applications, and thus can be explored as a potential solution for in-network processing for sensor-based applications [19].

The maturity of the cloud technology is providing a foundation for emerging data intensive applications in a variety of domains. The current technology trends, such as IoT, Social Media and Big Data, are utilizing the cloud based infrastructure services to support data life cycle from ingestion, analysis to storage. Building cyber infrastructure for such data intensive applications is a challenging issue. In this article we have discussed requirements and approaches concerning cyber infrastructures for sensor-based applications. The discussion was carried out from four different perspectives: sensor deployment and management, sensor provenance, sensor security and sensor data stream processing. We have discussed a few recent approaches and then outlined open research directions. We believe that current technology trends in IoT and embedded systems will further push the need for cyber infrastructures for sensor-based applications.

References

[1] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials*, 16(1), 414-454, 2014.

[2] L. Wang, and R. Ranjan. Processing Distributed Internet of Things Data in Clouds. *IEEE Cloud Computing* 1 (2015): 76-80.

[3] R. Ranjan. Streaming Big Data Processing in Datacenter Clouds. *IEEE Cloud Computing*, 1.1 (2014): 78-8

[4] A. Baggio. Wireless sensor networks in precision agriculture. *ACM Workshop on Real-World Wireless Sensor Networks* (REALWSN 2005), Stockholm, Sweden. 2005.

[5] S. Coyle, K.T. Lau, N. Moyna, D.O. Gorman, D. Diamond, F. Di Francesco, et al. BIOTEX—Biosensing textiles for personalised healthcare management. *IEEE Transactions on Information Technology in Biomedicine*, 14(2), 364-370, 2010.

[6] B. Shebaro, D. Midi, and E. Bertino: Fine-grained analysis of packet losses in wireless sensor networks. Proceedings of *Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking, SECON 2014*, Singapore, June 30 - July 3, 2014.

[7] S. Sultana and E. Bertino: A Comprehensive Model for Provenance. Invited Paper, Proceedings of *ER 2012 Workshops CMS, ECDM-NoCoDA, MoDIC, MORE-BI, RIGiM, SeCoGIS, WISM*, Florence, Italy, October 15-18, 2012.

[8] B. Shebaro, S. Sultana, S, R. Gopavaram, and E. Bertino: Demonstrating a lightweight data provenance for sensor networks. Proceedings of the *ACM Conference on Computer and Communications Security, CCS'12*, Raleigh, NC, USA, October 16-18, 2012.

[9] S. Sultana, G. Ghinita, E. Bertino, and M. Shehab: A Lightweight Secure Scheme for Detecting Provenance Forgery and Packet DropAttacks in Wireless Sensor Networks. *IEEE Trans. Dependable Sec. Comput.* 12(3): 256-269 (2015)

[10] S. R. Hussain, C. Wang, S. Sultana, and E. Bertino: Secure data provenance compression using arithmetic coding in wireless sensor networks. Proceedings of *the IEEE 33rd International Performance Computing and Communications Conference, IPCCC 2014*, Austin, TX, USA, December 5-7, 2014.

[11] J. S. Vitter, P. G. Howard, "Arithmetic coding for data compression," in *Information Processing and Management*, 1994, pp. 749–763.

[12] M. A. Rassam, M. A. Maarof and A. Zainal, A Survey of Intrusion Detection Schemes in Wireless Sensor Networks, *American Journal of Applied Sciences*, vol. 9, no. 10, pp. 1636-1652, 2012.

[13] S.H. Seo, J. Won, S. Sultana, and E. Bertino: Effective Key Management in Dynamic Wireless Sensor Networks. *IEEE Transactions on Information Forensics and Security* 10(2): 371-383 (2015).

[14] S. Al-Riyami and K. Paterson, Certificateless public key cryptography, *ASIACRYPT 2003*, Vol. 2894, pp. 452-473, 2003

[15] J. Habibi, A. Panicker, A. Gupta, and E. Bertino: DisARM: Mitigating Buffer Overflow Attacks on Embedded Devices. To appear in Proceedings of *9th International Conference on Network and System Security, NSS'15*, November 3-5, 2015,New York City, USA.

[16] S. Sultana, D. Midi, and E. Bertino: Kinesis: a security incident response and prevention system for wireless sensor networks. Proceedings of the *12th ACM Conference on Embedded Network Sensor Systems, SenSys '14*, Memphis, Tennessee, USA, November 3-6, 2014.

[17] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. SIGMOD Rec. 31, 3 (September 2002), 9-18. 2002

[18] U. Srivastava, K. Munagala, and J. Widom. Operator placement for in-network stream query processing. In Proceedings of the *twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS '05).* ACM, New York, NY, USA, 250-258.

[19] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13-16, 2012.

[20] L. M. Vaquero and L. Rodero-Merino. Finding your Way in the Fog: Towards a Comprehensive Definition of Fog Computing. *SIGCOMM Computing Communication Review* 44 (5): 27-32. 2014.