A Synchronized Shared Key Generation Method for Maintaining End-to-End Security of Big Data Streams

Deepak Puthal
University of Technology
Sydney, Australia
deepak.puthal@gmail.com

Surya Nepal, Rajiv Ranjan
CSIRO Data 61, Australia
Surya.Nepal@csiro.au,
rranjans@gmail.com

Jinjun Chen
University of Technology
Sydney, Australia
jinjun.chen@gmail.com

Abstract

A large number of mission critical applications ranging from disaster management to smart city are built on the Internet of Things (IoT) platform by deploying a number of smart sensors in a heterogeneous environment. The key requirements of such applications are the need of near real-time stream data processing in large scale sensing networks. This trend gives birth of an area called big data stream. One of the key problems in big data stream is to ensure the end-to-end security. To address this challenge, we proposed Dynamic Prime Number Based Security Verification (DPBSV) and Dynamic Key Length Based Security Framework (DLSeF) methods for big data streams based on the shared key derived from synchronized prime numbers in our earlier works. One of the major shortcomings of these methods is that they assume synchronization of the shared key. However, the assumption does not hold when the communication between Data Stream Manager (DSM) and sensing devices is broken. To address this problem, this paper proposes an adaptive technique to synchronize the shared key without communication between sensing devices and DSM, where sensing devices obtain the shared key re-initialization properties from its neighbours. Theoretical analyses and experimental results show that the proposed technique can be integrated with our DPBSV and DLSeF methods without degrading the performance and efficiency. We observed that the proposed synchronization method also strengthens the security of the models.

1. Introduction

There are a large number of critical applications, such as large-scale sensor networks for environment sensing, disaster management, remote health monitoring and smart homes, that require near real-time data stream to be processed in datacentres for enabling data-driven decisions. These applications produce high volume, velocity data that should be processed in near-real time to detect events such as heart-attacks in the context of remote health monitoring and telephony frauds in the context of telecommunication. These applications require a paradigm shift as compared to traditional store and process later approaches [1]. Clearly, traditional approaches cannot support near real-time decision making. To address this near real-time decision making requirement, a new cloud-based computing paradigm based on Stream Processing Engines (SPEs) has evolved [4, 17, 18]. SPEs can process the data stream on the fly [15, 16] in contrast to store and process later approaches enabled by batch processing engines such as Apache Hadoop and Amazon Elastic MapReduce. The need of real-time processing for high volume and high velocity input data arises due to the need of real time detection of events in combination with the fact that the data cannot be persisted for later analysis for practical reasons (e.g., data storage overhead) [12]. SPEs can process data in near real-time, but they have security limitations as discussed next. In addition, Data Stream Manager (DSM) undertakes the security verification of the data blocks on-the-fly before SPEs. These features present significant opportunities and challenges in the area of data security and freshness of big data stream [11, 12].

Let us consider a Disaster Management (DM) application to motivate the end-to-end (i.e., from Sensing Devices to Cloud Data Centre processing layer) security problem that exists with the current generation of SPEs and relevant stream processing algorithmic approaches. DM applications rely on near real-time processing of sensor data in the cloud. Efficiency and effectiveness of decision making and event (e.g., flooding, tsunami, landslides etc.) detection in DM applications is dependent on the following security related properties of the sensor data including confidentiality, integrity, authenticity and freshness. Any compromise on the above mentioned security related properties of data during processing and/or transmission will lead to inaccurate event detection and decision making. Ultimately this leads to the loss of lives and critical infrastructures. Hence, these applications require end-to-end security to increase the reliability of the data analysis pipelines.

URI: http://hdl.handle.net/10125/41889
ISBN: 978-0-9981331-0-2

StreamShield is a stream centric architecture for security and privacy in data stream environments, where authors highlighted the requirement of security in data stream for the very first time [2]. Authors broadly divided the security issues in two parts (i.e. data security and query security) and both these security issues were applied for stream data analysis. Following this architecture, we identify four important requirements and properties for security verification of big data stream: (a) near real-time security verification, (b) dealing with high volume and velocity of data, (c) the data items should only be accessed once, and (d) the original data are not available for comparisons [11, 12, 13, 14]. We focused on addressing big data stream security requirements by keeping all these big data stream constraints. We proposed a novel light weight security model by ensuring end-to-end security for big data streams. First, we proposed a Dynamic Prime Number Based Security Verification (DPBSV) scheme for big data stream processing, which is based on a common shared key that is updated dynamically by generating synchronized pairs of prime numbers [11, 12]. Later to make it more efficient and reduce the computational overhead and buffer size, we proposed a Dynamic Key Length Based Security Framework (DLSeF) based on the shared key derived from synchronized prime numbers [13, 14]. These two techniques were proposed to maintain the end-to-end security of big sensing data stream and perform security verification at DSM.

All these above security solutions follow the independent rekeying process without further communications between the source sensors and DSM after handshaking. However, it is impossible to continue the key update and data transmission without any interruptions in a hostile nature of source sensing area. Hence, the source side key generation synchronization is a major problem with above security solutions (DPBSV and DLSeF). In these models, a source node sends a request message to DSM to get the synchronization properties if there is any kind of key desynchronization, which is not an efficient way to get synchronization properties. By focusing on this problem, we propose a novel synchronization method in this paper in which source sensors get synchronization properties from their neighbours. As the sensing sources are distributed in a hostile environment and nodes do not have any neighbour/network information, it is a challenging task to identify the authenticated neighbour and retrieve the synchronization properties. The contributions of the paper is summarized as follows:

- We present a synchronized shared key generation method.
- We apply the synchronization method over DPBSV and DLSeF security architecture.
- We evaluate the model both theoretically and empirically.

The rest of this paper is organized as follows. Section 2 details the related literature. Section 3 presents the problem statement for key synchronization in the big data stream based applications. Section 4 describes the proposed synchronization method and its association with DLSeF architecture. Section 5 presents the theoretical analysis, and section 6 evaluates the performance and efficiency of the method through experiments. Section 7 concludes the paper by outlining potential future works.

2. Related works

Ensuring the end-to-end security of big data streams has emerged as an important research topic in many stream processing applications such as disaster management, telephony fraud detection and credit card fraud detection. In this section, we describe related works that cover the research areas such as stream processing, and secure authentication of neighbours.

Stonebraker et al. [1] outlined eight core features that a framework or system must possess in order to be able to efficiently handle stream processing workloads. These core features include (i) the continuous flow nature of data stream, (ii) handling the data imperfection, (iii) maintaining the data security, (iv) integrated store and stream data, (v) partition and scale applications automatically, (vi) query processing on streams, (vii) expectations of query outcomes and (viii) process and respond instantaneously. Our aim is to ensure the data safety (iii) and availability (viii) features of big data streams. Nehme et al. initially proposed an architecture by addressing the needs of data security and query security in streaming environments [2]. They proposed a continuous access control architecture, named StreamShield, which ensures query security. However, StreamShield is unable to ensure end-to-end data security of streams.

Arasu et al. proposed a Data Stream Management System (DSMS), called STanford stREam data Manager (STREAM) [4]. It is intended to deal with high velocity data rates and substantial numbers of continuous queries through adaptive resource allocation; however, STREAM cannot ensure data security properties. Similar to STREAM, StreamCloud is a large scalable elastic data streaming system for processing large data stream in cloud [3].

Sung et al. describes an identification based node authentication, which can be used to solve key agreement problem in a three-layer interaction of sensor networks [8]. Authors consider the characteristics, architecture, and vulnerability of the sensors, and provides an ID-based node authentication scheme. An Elliptic Curves Cryptography (ECC) based

authentication protocol has been proposed in WSN [9] for device authentication. Khan et al. [10] showed the M.L. Das-scheme's security pitfalls and proposed an improvement and security patches that attempt to fix the susceptibilities of this scheme. Both of the above are a secure authentication protocol but are computationally rich, which do not follow the big data stream properties as stated in the Introduction section. Park [24] explained an interesting technique to get the time stamp from neighbours for robustness to clock skews among nodes. The neighbour is authenticated to get the synchronization properties. In [25], a sensor classified its neighbours based on the geographic location and communicated to the trusted neighbours. Our network structure is different from other network structure by considering DSM as a centralized processor; so we need a new solution for our model. In this paper, we propose a new synchronization method on the above DPBSV and DLSeF models.

3. Problem statement

Security of big sensing data stream is a major issue for several applications including disaster management, emergency management, event detections etc. [12]. By considering these applications, DPBSV and DLSeF are two security solutions proposed to maintain the end-to-end security in big sensing data stream. In both of these models, source sensor devices and DSM never communicate between themselves after handshaking. During handshaking process, DSM sends the key generation properties to source sensors and the sensors save the sensitive key generation properties in secure module of the sensor such as TPM. The TPM is a dedicated security chip following the Trust Computing standard specification for cryptographic microcontroller systems [23]. TPM provides a cost effective way of "hardening" many recently deployed applications, those are previously based on software encryption algorithms with keys kept on a host's disk [19, 23]. It provides a hardware based trust, which contains cryptographic functionality like key generation, store, and management in hardware. So source sensor performs the rekeying process independently. The above security solutions follow these methods to satisfy the big data stream properties (from Introduction Section). Here, synchronization in shared key generation between the source and DSM is a major issue and needs to be solved. The complete architecture of data flows from source sensing device to cloud datacenter with possible attacks is shown in Figure 1. We refer to [8] for further information on stream data processing in cloud.

To address these issues and make the security solutions more efficient, we propose a novel synchronization technique for big data stream. Authentication of neighbour nodes to get the keys, clock skew, and other properties are very common for wireless sensor networks [24, 25]. We use a similar method with minor modifications according to our network structure to get the key synchronization properties from neighbours. Different network structures use different properties such as cluster head or group key or base station information, for neighbour node authentication [26, 27]. According to our network structure, all sources have DSM properties along with the current time interval. So we use these properties for neighbour authentication. We follow the method to get the properties from neighbours because all source sensors use the same key in the given time interval to perform the encryption. There are two major synchronisation issues that need to be addressed for DPBSV and DLSeF model: (a) time synchronization (follow particular time to start the key generation process), and (b) the synchronisation of the shared key when source sensor missed the current key because of a malicious activity or natural hazards.

As the source sensing area is distributed in nature and the source performs the shared key generation independently, time to start the key generation is a challenging and important issue for security models. In any hazardous situations, sensors may miss the shared keys or key synchronization. Because of the TPM properties, key generation properties remain safe in sensors. So sensors only need the key generation properties to restart synchronize key generation and send data blocks to DSM.

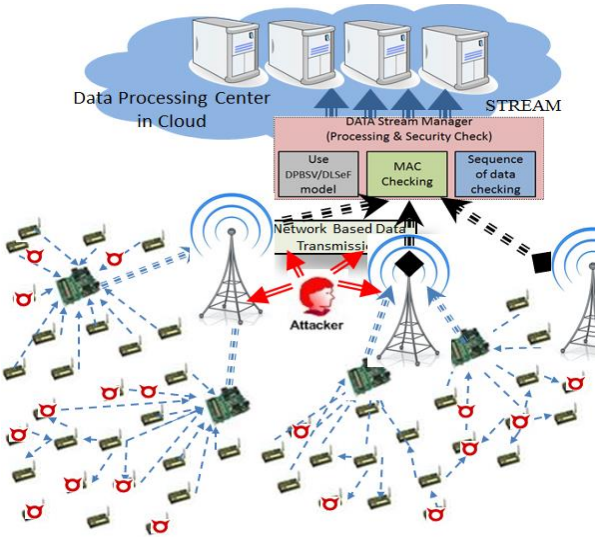


Figure 1. Overlay architecture of sensing device to cloud data processing center, and possible attacks during data flow.

4. Proposed synchronization method

Our security model is motivated by the concept of moving target defense. The basic idea is that the keys

are the targets of attacks by attackers. To avoid such problems in big sensing data streams, we proposed novel techniques such as DPBSV [12] and DLSeF [14]. In these models, if an intruder/ attacker eventually hacks the key, the data and time period is selected in such a way that he/she cannot predict the key or its length for the next session. In such models, there are two major synchronisation issues that need to be addressed: (a) the precise time to start the key generation process (time synchronization) and (b) the synchronisation of the shared key as discussed before. While addressing the synchronisation issues, it is important to note that no compromise is made on the authenticity, integrity and partial confidentiality (maintain confidentiality in real time) of the data, which are important to make decision from the collected data. In this paper, we have addressed the initial process synchronization properties with the lost shared key synchronization over DLSeF model.

Table 1. Notations used in our model

Acronym	Description
S_i	i^{th} source sensing device's ID.
D_i	DSM ID
K_i	i^{th} source device's secret key.
K_{si}	i^{th} source device's session key.
kl	Key length
K_{SH}	Secret shared key at sensor and DSM
K_{SH^-}	Previous secret shared key.
r	Pseudo random number.
t	Interval time to generate the prime number.
T	Timestamp added with data blocks.
T'	Current time
T''	Time to start the process.
P_i	Random prime number.
K_d	Secret key of the DSM.
I_D	Data for integrity check.
A_D	Secret key for authenticity check.
$E( )$	Encryption function.
$H( )$	One-way hash function.
$Prime(P_i)$	Prime number generation function.
KeyGen	Key generation procedure.
Key-Length ( )	Key length selection procedure.
$\oplus$	Bitwise X-OR operation.
$\|$	Concatenation operation.
RQA	Authentication request message.
RPA	Authentication response message.

Similar to DPBSV and DLSeF security solutions [11, 12, 13, 14], we added synchronisation processes (both time synchronization and key synchronization) to them with the standard steps: system setup, handshaking, rekeying, key synchronization and security verification. We follow DLSeF system setup, handshaking, rekeying model with minor modifications before synchronization properties are described. Table 1 provides the notations used in modelling our method. We next describe the proposed method.

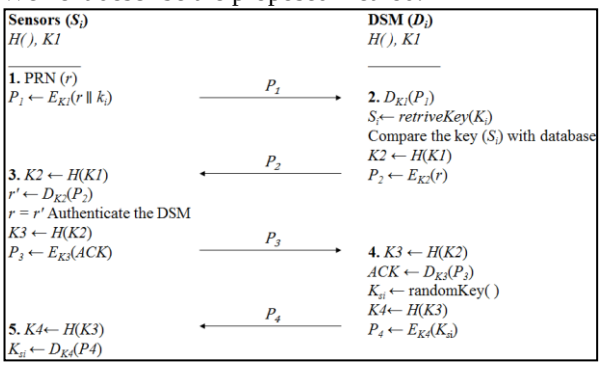


Figure 2. Secure authentication between Sensor and DSM during system setup (from DLSeF model [13]).

4.1. System setup

In this step, we assume that DSM has all deployed sensing device's identities (IDs) and respective secret keys because the network is untrusted and hostile in nature. Sensing devices and DSM implement some common primitives such as hash function (i.e. H( )), and common key (*K1- K4*), which are executed during the initial identification and system setup steps.

The proposed authentication process follows the DLSeF authentication phase that includes five different steps [13, 14]. The first three steps are for the sensing device and DSM authentication process and the final two steps are for the session key generation process as shown in Figure 2. The session key (K_{si}) is utilized during the handshaking process which was generated during the system setup step.

We keep the hashing and shared key at the source sensor to use in future for data encryption and neighbour authentication (refer Figure 2). We are using the trusted part of sensors (i.e. TPM) to keep the secret information of source sensors [19].

4.2. Handshaking

In the handshaking process, the DSM sends the key generation and synchronization properties to sensors based on their individual session key (K_{si}) established earlier during authentication process.

The dynamic prime number generation function computes the relative prime number, which always depends on the previous prime number [13]. It is also already proved that the generated number will always be prime number and synchronized between source devices and DSM [13]. We follow DLSeF method for rekeying time interval according to the key length.

DSM sends a time chunk, i.e., T'' along with other properties i.e. $K_d, t, , P_i,$

$Prime\ (), KeyLength(), K_{SH}, KeyGen$ [11, 12, 13, 14]. This time stamp (T'') is used at source to initialize the key generation process and sent the encrypted data blocks to the DSM. If any sources missed the time stamp to initialize the process, it will send request to DSM to get the time stamp again. New sources joining to the network need to follow the step to start the key generation/ rekeying process.

$S_i \leftarrow$ DSM: $\{E_{K_{Si}}(K_d, t, T'', P_i, Prime\ (), KeyLength(), K_{SH}, KeyGen)\}$

All of these above transferred information are stored in the trusted part of source for future rekeying process (e.g., TPM) [19].

Table 2. Time taken by symmetric key (AES) algorithm to get all possible keys using the most advanced Intel i7 processor.

Key Length	32	64	128
Key domain size	4.295e+09	1.845e+19	3.4028e+38
Time (in nanoseconds)	7.301e+09	3136e+19	5.7848e+35

4.3. Rekeying

Our proposed method not only calculates the dynamic prime number to update the shared key without further communication after handshaking, but also dynamically change the key length at sensor and DSM. We follow the DLSeF Rekeying process to ensure that the protocol remains secured [13]. According to the properties of the TPM, no one have access to contents which is stored inside the TPM. Only the corresponding sensor can access TPM properties [19]. From the *Handshaking* process, sensors are aware of the *Prime* (*Pi*), *KeyLength*, and *KeyGen*. Now we describe the complete rekeying process by using those functions and keys from DLSeF model. The synchronized dynamic prime number *Pi* is generated on both ends, i.e., sensors and DSM [13], to be used for the rekeying process. Now sensors need to wait for the time T'' to start the key generation process.

ALGORITHM 1. Key Generation (Rekeying) Process

1. Dynamic prime number *Prime* (P_i) [13].
2. Following DLSeF method [13]:
   2.1 t (time interval) = {t1, t2, t3, …}
   Here t1, t2, t3, … are the time interval for rekeying. (*32/64/128-bit key from DLSeF*)
   2.2 At time (t), S_i and D compute K_{SH}= $E_{K_{SH}}(H(P_i, K_d))$.
   2.3 After time (t), reinitialize from *Step 1*.
3. The encryption process at sensor as follows
   3.1 $I_D=DATA \oplus K_{SH}$ // For integrity check
   3.2 $A_D=S_i \oplus K_{SH}$ // For Authentication check
4. $S_i \rightarrow$ DSM: $\{(I_D \| (A_D \| T))\}$// Data format to DSM

By following DLSeF model, sensors generate the shared key $K_{SH}=(E(Pi,Kd))$ using the prime number Pi, and DSM's secret key $E(Pi,Kd)$. We use the secret key of DSM to improve the robustness of the security verification process and fixed the initial key length as 64 bits. The data blocks divided into two different parts, i.e., authentication and integrity verification. One is encrypted DATA based on shared key K_{SH} for integrity checking (i.e., $ID=DATA \oplus KSH$), and the other part is for the authenticity checking (i.e., $AD=S_i \oplus K_{SH}$). The resulting data block $((DATA \oplus K_{SH}) \| (S_i \oplus K_{SH}))$ is sent to DSM as follows:

$S_i \rightarrow$ DSM: $\{(ID \| (AD \| T))\}$.

The procedure of rekeying process is shown in Algorithm 1.

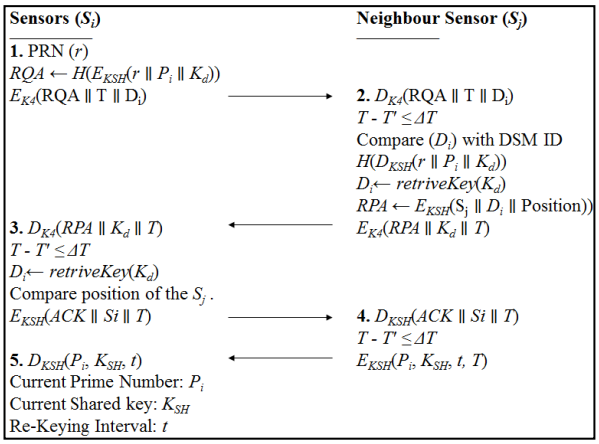


Figure 3. Neighbour node discover to get the current state of key generation properties.

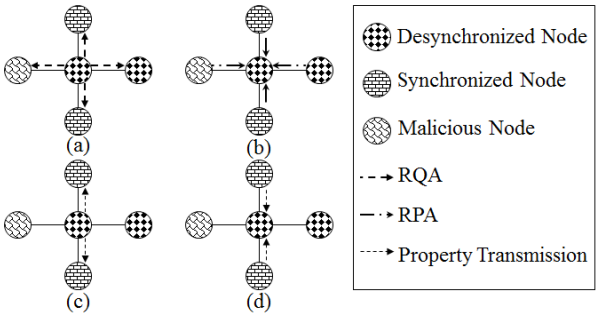


Figure 4. Neighbour discovery to get the key synchronization properties with all possible conditions. (a) node S_i sends RQA message to all its one-hop neighbours; (b) the sender receives the RPA of individual RQA; (c) S_i send ACK to only authenticated synchronized neighbours; (d) node S_i receives the synchronization properties.

4.4. Key synchronization

Synchronization is one of the major issues during the rekeying process between sensors and DSM, as they are

not interacting after handshaking process. The shared key synchronization is based on the initial key generation process followed by the rekeying. So the initial key synchronization is to make a common time to start the key generation process. In our model, DSM works as a centralize controller. So DSM initiates the key generation process. As defined before, during the handshaking process DSM sends back to the source (S_i) with a time stamp T'' to initialize the key generation process.

There are potentially two cases (i) sensor starts the process on time to maintain synchronization; (ii) sensor may be missing the time stamp T'' or later receives the key generation properties after time stamp. In the second case, source sensor send request to get the next time stamp for key generation process.

There are several reasons for sensors to be out of sync such as inability of the source node to generate the shared key by some computational overhead or by any natural disaster or by any malicious activity. Even if a sensor missed the synchronization, it does not miss the key generation properties because of the TPM features [19]. In such cases, the source sensor (S_i) gets synchronization properties from its neighbours. According to the source network structure, sensors do not have neighbour information. So it's a challenging task to identify the neighbours and get the key synchronization properties. The procedure to obtain shared key properties from unknown neighbours is given below.

4.4.1. Initial setup. Let us assume that sensor (S_i) missed the synchronization. The Sensor (S_i) computes a Pseudo Random Number, i.e., $PRN(r)$, using the current prime number (P_i) and the shared key (K_{SH}) to generate the authentication request message (RQA) i.e. $RQA \leftarrow H(E_{KSH}(r \mathbin{//} P_i \mathbin{//} K_d))$. Then the resultant RQA, DSM ID (D_i) and time stamp (T) encrypt with mutual key K4 from the system setup steps ($E_{K4}(RQA \parallel T \parallel D_i)$) (refer to Figure 2). We use this key for encryption because all the authenticated nodes have this key from DSM during the system setup phase.

4.4.2. Synchronization phase. The out of sync sensor (S_i) broadcasts this to its one-hop neighbours. When the neighbour sensors receive the information, it decrypts with its mutual key i.e. K4 ($D_{K4}(RQA \mathbin{//} T \mathbin{//} D_i)$). It compares the received time frame (T) with its current time (T') to check the data freshness and avoid the replay attack ($T - T' \leq \Delta T$). If the time difference is less then ΔT, then it accepts the data packet; otherwise it is discarded. Here ΔT is the average time required to transmit data packet between source and DSM.

The neighbour node (denoted as S_j) compares the received DSM ID with its own DSM ID to validate the source as the authenticated one. To make the authentication process stronger, we perform two layer encryption of the request (RQA). Sensor (S_j) perform the hash and decrypt the second layer with the shared key (K_{SH}), i.e. $H(D_{KSH}(r \mathbin{//} P_i \mathbin{//} K_d))$. It uses previous shared key if the shared key K_{SH^-} is modified in the meantime and compares the DSM ID by retrieving it using the DSM secret key ($D_i \leftarrow retriveKey(K_d)$).

After authentication process, S_j prepares authentication response message (RPA) by including its own ID, DSM ID and pseudo random number r ($RPA \leftarrow E_{KSH}(S_j \mathbin{//} D_i \mathbin{//} r)$). It then encrypts the RPA along with DSM key and time stamp by using the same key K4 ($E_{K4}(RPA \mathbin{//} K_d \mathbin{//} T)$).

Once S_i receives the RPA, it is processed in the same way to authenticate the node S_j ($D_{K4}(RPA \parallel K_d \parallel T)$). First it compares the time to avoid replay attack ($T - T' \leq \Delta T$) and compares the DSM ID ($D_i \leftarrow retriveKey(K_d)$) and value of r to perform authentication. Here desynchronized source node (S_i) encounters three different types of neighbours: malicious node, desynchronized authenticated node and synchronized authenticated node as shown in Figure 4. Malicious neighbours cannot decrypt S_i request because it is encrypted by the secret key. But a desynchronized authenticated node can read the request. Once it came to know that the source (S_i) is seeking the key synchronization properties, it sends the response with its desynchronization indication. The source discards the RPA received from such nodes. If the source node receives RPA from authenticated synchronised neighbour, S_i choses such node by sending the ACK in order to get the key synchronization properties ($E_{KSH}(ACK \parallel Si \parallel T)$).

This acknowledgement message (i.e. ACK) confirms the mutual authentication between the source and synchronised neighbour to obtain the key synchronization properties ($D_{KSH}(ACK \parallel Si \parallel T)$). After receiving the acknowledgement message, the authenticated neighbour gets the source node ID and sends the shared key properties (P_i, K_{SH}, t) to source node as $E_{KSH}(P_i, K_{SH}, t, T)$.

When the desynchronized source gets the shared key synchronization properties ($D_{KSH}(P_i, K_{SH}, t, T)$), it can generate the shared key by itself, because it has the prime number (P_i), shared key (K_{SH}), and time to change the next key (t). Every time we are checking the time interval in order to avoid the replay and DoS attacks. The stepwise representation of the neighbour authentication to obtain the shared key properties is shown in Figure 3.

4.4.3. New node synchronization. If there is a new source node joining to the network, then it starts the authentication process with DSM to get the key generation properties. After receiving the key generation properties from DSM, the node (n) either

starts the process or authenticate with the neighbour nodes to compare the synchronization properties.

4.5. Security verification

In this step, the DSM first checks the authenticity in each individual data block AD and then the integrity with the randomly selected data blocks ID. Here data block is divided into two blocks for authenticity checking and integrity checking. Along with authenticity checking, we add timestamp (T) in order to get the data freshness and avoid replay attack. We change the security verification for data integrity in random interval of data packets according to the DLSeF properties [13, 14]. We prefer to change the integrity verification interval that is directly proportional to the shared key length because the key length is inversely proportional to the possibilities of data accessible. The data block at DSM for security verification is represented as: $\{(ID\|(AD\|\mathrm{T}))\}$. DSM first checks the authentication part to get the timestamp. It compares its own timestamp with the received one i.e. $T - T' \leq \Delta T$. If the time interval is less than or equal to the predefined time ΔT, then it accepts the data; otherwise it is rejected. This will help to maintain the data freshness and avoid the replay attack. Initial time checking and the authenticated source checking can avoid the DoS (denial of service) attack. Another important advantage of adding the time stamp (T) is to get the shared key used for the encryption process. If the shared key is updated after receiving the data block encryption, then DSM uses the previous shared key (K_{SH^-}) for decrypting the data instead of current key (K_{SH}).

We are updating the shared key before the possible attacks. For the authenticity check, the DSM decrypts AD with shared key $Si=AD \oplus KSH$. Once Si is obtained, the DSM checks its source database and extracts the corresponding secret key Ki ($K_i \leftarrow retrieveKey(S_i)$). In the integrity check process, the DSM decrypts the selected data such as $DATA=ID \oplus KSH$ to get the original data and checks MAC for the data integrity.

5. Theoretical analysis

This section provides a theoretical analysis of our proposed model to show that the proposed synchronization method works efficiently by getting the shared key properties from the neighbours. We also apply the synchronization properties over DPBSV and DLSF and prove that the models are safe against the network attacks. Proposed synchronization method never interrupt the shared key generation at sensors.

We have made a number of practical and realistic assumption in our method. In the following, we first describe those assumptions.

Assumption 1. In our method, the data that was encrypted by a symmetric-key algorithm cannot be decrypted by any parties, unless they have the session/shared key.

Assumption 2. Shared key (K_{SH}) calculation procedures reside inside trusted parts of the sensor (like TPM) so that no one is authorized to access and manipulate them [12].

We define our threat model, which is similar to the most cryptologic analyses, to the shared key properties as follows:

Theorem 1. According to the proposed synchronization method, the shared key (K_{SH}) is always synchronized between Source sensor (Si) and DSM.

Proof: We are following the DLSeF security verification model and added the shared synchronization properties to it. According to DLSeF properties, the dynamic shared key length varies in 32 bit, 64 bit, and 128 bit; these keys are updated both source and DSM ends. The shared key is updated without further communications between S_i and DSM after handshaking. A variation in key length introduces a complexity to the attackers to predict the next shared key. The ECRYPT II recommendations on key length say that a 128-bit symmetric key provides the same strength of protection as a 3,248-bit asymmetric key. Advanced processor (Intel i7 Processor) took about 1.7 nanoseconds to try out one key from one block. With this speed, it would take about 1.3 × 1012× the age of the universe to check all the keys from the possible key set [22]. All the related key domain and the time required to get the possible keys by using Inter i7 processor are listed in Table 2. We follow the DLSeF model to select the key lengths [13].

Here, we are highlighting the synchronization in two places (i) source sensor with DSM at initial key generation process and (ii) while obtaining the synchronization properties from neighbour. For the first option (during the handshaking process), DSM sends the key generation properties to S_i along with the timestamp (T'') to set the key generation time. Then both DSM and S_i generate the shared key with dynamic length and interval as in DLSeF method. This means the shared key will be synchronized at both ends. In second option (obtaining the synchronization properties from neighbours), if any of the source desynchronized, it initiates the neighbour authentication process to discover authenticated synchronized neighbour (see Figure 3). After authentication, neighbour sends the key generation properties $E_{KSH}(P_i, K_{SH}, t, T)$, where T is for data freshness and t is the start of the key generation process. Then source S_i can use the current key and use these properties to update the next key (i.e. $K_{SH}=(E(Pi,Kd))$) after time t. Now source S_i became synchronize with other sources and DSM.

Theorem 2. After applying synchronization, security verification models (DPBSV and DLSeF) are protected against authentication, integrity and partially confidentiality.

Proof: Please refer to [11, 12] for the attack properties associated with DPBSV model and [13, 14] with the DLSeF model. By considering TPM properties, we know that an attacker cannot get the secret information (P_i, K_i, K_{SH}) or the key generation properties (KeyGen). During the neighbourhood authentication procss, a sensor (S_i) shares the synchronization properties after authentication and gets the DSM ID and the secret key (see Figure 3). So there are no possibilities for the malicious nodes to trap authenticated sensors to get the shared key generation properties. Following neighbour synchronization properties, malicious nodes cannot interfere because neighbours identify each other through the DSM ID (K_d) and the encryption process uses the secret key (E_{K4}). Those properties are not known to malicious nodes. We know that an intruder cannot get the currently used K_{SH} within the time interval t (see Table 2), because our proposed method calculates P_i randomly after time interval t and then uses the value P_i to generate K_{SH}. But an attacker can introduce itself as an authenticated node to send packets.

We know that DPBSV [11] and DLSeF [13] are protected against authentication, integrity and partially confidentiality. From above, we conclude that, an attacker cannot get the shared key information during neighbour synchronization. By combining the above two we conclude that the security verification models are safe after including the synchronization properties.

Theorem 3. After applying the synchronization, the security verification models avoid replay attacks.

Proof: There are potentially two places for replay attacks (i) during the neighbour authentication; (ii) the security verification at DSM. In both of these cases we are adding a time stamp i.e. T in packets. During the security verification at DSM, DSM checks for the data freshness by comparing the time interval between the sent and received time of data blocks such as $T - T' \leq \Delta T$. If the interval is less than or equal to ΔT, then the data block is accepted; otherwise it is rejected. Application of this rule keeps rejecting the delayed data packets. but maintains the data freshness and avoids the replay attacks. Through the time interval (ΔT), it is easy for DSM to find the shared key used for encryption ($K_{SH}-$ or K_{SH}). We also follow the same method to avoid replay attack during neighbour authentication. By using such method, our model is proven to be more efficient to avoid the DoS attacks.

6. Experiment and evaluation

In order to evaluate the performance of the proposed key synchronization method under the adverse conditions, we validate our proposed method in a well-established security protocol simulation environment. We first verify the security method using Scyther [5], and then measure the efficiency of the same in the JCE (Java Cryptographic Environment) [6]. Finally, we check the performance of security aware sensor data encryption and sensor node performance in COOJA simulator provided by Contiki OS [7].

6.1. Security verification

The proposed method for synchronized shared key is implemented in the Scyther simulation environment using the Security Protocol Description Language (.spdl). The efficacy of the proposed security is observed for two important instances (i) during the security verification at DSM and (ii) during neighbour authentication process. According to the features of Scyther, Si is the sender (i.e., source sensor), Sj is the neighbour of Si (neighbour authentication) and D is the recipient (i.e., DSM). Apart from these, we follow the default properties of Scyther. Many types of cryptographic attack can be considered in our simulation context. In our case, we focus on integrity, authentication, data confidentiality (in real time), and replay attacks as discussed above. We used Scyther, an automatic security protocols verification tool, for verifying our model.

Scyther results : verify

Claim				Status		Comments
Auth	Si	Auth,Si2	Secret Di	Ok	Verified	No attacks.
		Auth,Si3	Secret Kd	Ok	Verified	No attacks.
	SJ	Auth,SJ2	Secret Di	Ok	Verified	No attacks.
		Auth,SJ3	Secret Kd	Ok	Verified	No attacks.

Figure 5. Secure authentication results.

Scyther results : verify

Claim				Status		Comments
MS	Si	MS,Si2	Secret Pi	Ok	Verified	No attacks.
		MS,Si3	Secret KSH	Ok	Verified	No attacks.
	D	MS,D2	Secret Pi	Ok	Verified	No attacks.
		MS,D3	Secret KSH	Ok	Verified	No attacks.

Figure 6. Security verifications results at DSM.

Results: We did our simulation using variable numbers of data blocks in each run. Our experiment ranges from 10 to 100 instances with the intervals of 10. During the neighbour authentication, both sensors Si and Sj authenticate themselves while hiding the DSM ID and secret key. In the experiment, we did not encounter any attacks that can compromise the security

properties of the big data streams. Results shown in Figure 5 validate the above hypothesis; it also and shows the neighbour authentication in the Scyther environment. As stated in [13], we perform the security verification at DSM; here, we follow the same concept while adding the new key synchronization process. Figure 6 shows the results of the security verification at DSM after combining the synchronization method with DLSeF.

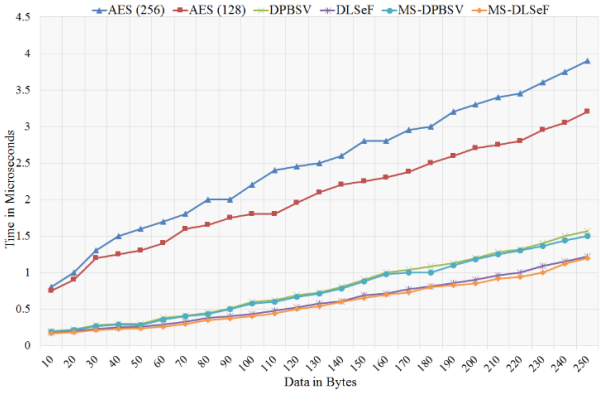


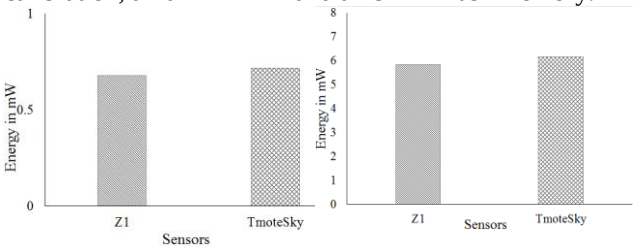
Figure 7. Performance of security verification at DSM.

6.2. Performance comparison

The performance comparison experiment is carried out in JCE (Java Cryptographic Environment). The performance is based on the features of JCE in java virtual machine version 1.6 64 bit. JCE is the standard extension to the java platform which provides a framework implementation for cryptographic methods. We have performed experiments with different sizes of data blocks by applying the synchronization over DLSeF and named as MS-DLSeF (DLSeF modified for synchronization). We also applied the same synchronization proprieties over DPBSV and named as MS-DPBSV (DPBSV modified for synchronization). The results of our experiments are shown in Figure 7. We compare the performance of our proposed method over DLSeF (MS-DLSeF) and DPBSV (MS- DPBSV) with the advanced encryption standard (AES) [20, 21], DPBSV [11, 12] and DLSeF [13, 14]. Apart from the neighbour synchronization we follow the properties of DPBSV and DLSeF. From Figure 7, it is clear that the synchronization method does not degrade the performance of DPBSV and DLSeF in terms of security verification speed.

6.3. Sensor node performance

We experimented with the performance of the sensors in terms of the overheads involved while computing synchronized shared keys in COOJA simulator provided by Contiki OS [7]. We modelled the two most common types of sensor, i.e., Z1 and Trotsky. A Z1 sensor node is equipped with the low power microcontroller MSP430F2617, which features a powerful 16-bit RISC CPU @16 MHz clock speed, built-in clock factory calibration, 8 KB RAM and a 92 KB Flash memory. TmoteSky is an ultra-low power sensor, and it is equipped with the low power microcontroller MSP430F1611, which has a built-in clock factory calibration, a 10 KB RAM and a 48 KB Flash memory.



(a) Energy for neighbour authentication **(b) Energy for security verification**

Figure 8. Energy consumption by using COOJA in Contiki OS.

In this experiment, we measured the performance of sensors while they transmit/receive information from neighbours or dynamically update the shared key for undertaking security verification process. Figure 8 (a) shows the energy required by sensors during transmitting/receiving synchronization properties from neighbours and Figure 8 (b) shows the power consumption behaviours for the key generation process. From these experiments, we conclude that our proposed method is lightweight as both the application of synchronization properties and security verification model consume very little sensor battery power.

From the above experiments, we conclude that our proposed method is secured and efficient in term of security verification speed.

7. Conclusion and future works

In this paper, we proposed a shared key synchronization method to ensure an end-to-end security in big data stream processing system consisting of distributed sensors and cloud-hosted stream processing engines (DSM). The proposed synchronization technique was implemented and verified in our previously proposed DPBSV and DLSeF security verification method for big data streams. In these previous models, sensors and DSM update their shared key independently without requiring further communication after handshaking phase. Proposed method synchronize the shared key without communication between sensing devices and DSM, where sensing devices obtain the shared key re-initialization properties from its neighbours. By

theoretical analyses and experimental evaluations, we showed that our proposed synchronization method successfully scales within the DPBSV and DLSeF models. In our future work, we will implement the proposed model in a real IoT application that requires near real-time decision making. We will further improve our techniques to meet the requirements of dynamic IoT networks.

8. Acknowledgements

This research is funded by an Australia India strategic research grant, titled “Innovative Solutions for Big Data and Disaster Management Applications on Clouds (AISRF-08140),” from the Department of Industry, Australia. The research in this paper is also partially supported by ARC LP140100816.

9. References

[1] M. Stonebraker, U. Çetintemel, and S. Zdonik. "The 8 requirements of real-time stream processing." *ACM SIGMOD Record*, 34(4), 2005, pp. 42-47.

[2] R. Nehme, et al. "StreamShield: a stream-centric approach towards security and privacy in data stream environments." In *Proc. of ACM SIGMOD International Conference on Management of data*, 2009, pp. 1027-1030.

[3] V. Gulisano, et al. "Streamcloud: An elastic and scalable data streaming system." *IEEE Transactions on Parallel and Distributed Systems*, 23(12), 2012, pp. 2351-2365.

[4] A. Arasu, et al. "STREAM: the stanford stream data manager (demonstration description)." In *Proc. of ACM SIGMOD international conference on Management of data*, 2003, pp. 665-665.

[5] Scyther,[Online] http://www.cs.ox.ac.uk/people/cas.cremers/scyther/

[6] M. Pistoia, et al. "Enterprise Java 2 Security: Building Secure and Robust J2EE Applications." Addison Wesley Professional, 2004.

[7] Contiki OS: http://www.contiki-os.org/ (accessed on: 04.08.2015).

[8] S. Sung, and J. Ryou. "ID-based sensor node authentication for multi-layer sensor networks." *Journal of Communications and Networks*, 16(4), 2014, pp. 363-370.

[9] H. Yeh, et al. "A secured authentication protocol for wireless sensor networks using elliptic curves cryptography." *Sensors* 11(5), 2011, pp. 4767-4779.

[10] M. Khan, and K. Alghathbar. "Cryptanalysis and security improvements of ‘two-factor user authentication in wireless sensor networks’." *Sensors* 10(3), 2010, pp. 2450-2459.

[11] D. Puthal, et al. "DPBSV--An Efficient and Secure Scheme for Big Sensing Data Stream." In *Proc. of Trustcom/BigDataSE/ISPA*, vol. 1, pp. 246-253. IEEE, 2015.

[12] D. Puthal, et al. "A Dynamic Prime Number Based Efficient Security Mechanism for Big Sensing Data Streams." *Journal of Computer and System Sciences* 83(1), 2017, pp. 22-42.

[13] D. Puthal, et al. "A Dynamic Key Length Based Approach for Real-Time Security Verification of Big Sensing Data Stream." In *Proc. of* 16th International Conference on Web Information System Engineering *(WISE)*, 2015, pp. 93-108.

[14] D. Puthal, et al. " DLSeF: A Dynamic Key Length based Efficient Real-Time Security Verification Model for Big Data Stream." In *ACM Transactions on Embedded Computing Systems (TECS)*, 2016.

[15] D. Carney, et al. "Monitoring streams: a new class of data management applications." In *Proc. of 28th international conference on Very Large Data Bases*, 2002, pp. 215-226.

[16] S. Chandrasekaran, et al. "TelegraphCQ: continuous dataflow processing." In *Proc. of ACM SIGMOD international conference on Management of data*, 2003, pp. 668-668.

[17] E. Bertino, S. Nepal, and R. Ranjan. "Building Sensor-Based Big Data Cyberinfrastructures." *IEEE Cloud Computing* 2(5), 2015, pp. 64-69.

[18] R. Ranjan. "Streaming big data processing in datacenter clouds." *IEEE Cloud Computing* 1(1), 2014, pp. 78-83.

[19] S. Nepal, et al. "A mobile and portable trusted computing platform." *EURASIP Journal on Wireless Communications and Networking*, 2011(1), 2011, pp. 1-19.

[20] PUB, NIST FIPS. "197: Advanced encryption standard (AES)." *Federal Information Processing Standards Publication* 197, 2001, pp. 441-0311.

[21] S. Heron. "Advanced Encryption Standard (AES)." *Network Security* 2009(12), 2009, pp. 8-12.

[22] www.cloudflare.com (accessed on: 04.08.2015).

[23] TCG Trusted Platform Module (TPM) specification, https://www.trustedcomputinggroup.org/specs/tpm/ (accessed on: 04.08.2015).

[24] T. Park. "LiSP: Lightweight Security Protocols for Wireless Sensor Networks." *PhD dissertation, The University of Michigan*, 2005.

[25] RA Shaikh, S. Lee, and A. Albeshri. "Security completeness problem in wireless sensor networks." *Intelligent Automation & Soft Computing* 21(2), 2015, pp. 235-250.

[26] MA Jan, et al. "A Sybil attack detection scheme for a forest wildfire monitoring application." In *Future Generation Computer Systems*, 2016.

[27] D. Puthal, et al. "Threats to Networking Cloud and Edge Datacenters in the Internet of Things." *IEEE Cloud Computing* 3(3), 2016, pp. 64-71.