Study on Reliable and Secure Multi-receiver Data Delivery for Internet of Things

January 2018

Ei Khaing Win

Study on Reliable and Secure Multi-receiver Data Delivery for Internet of Things

Submitted to
Graduate School of Information Science and Technology
Osaka University

January 2018

Ei Khaing Win

List of Publications

1. Journal Paper

1. Ei Khaing Win, Yoshihisa, T., Ishi, Y., Kawakami, T., Teranishi, Y., and Shimojo, S.: A lost sensor data recovery scheme for sensor data stream multicasting. In *J. of Information Processing*, Vol. 26, 2018 (to appear).

2. International Conference Paper (with review)

1. Ei Khaing Win and Mie Mie Su Thwin.: SEC: Security control system for the integration of mobile agent and web services based on certificateless cryptography and two party hashing key agreement protocol. In *Proc. of 11th Intl. Conf. on Computer Application (ICCA 2013)*, pp. 553–557, 2012.

2. Ei Khaing Win, Yoshihisa, T., Ishi, Y., Kawakami, T., Teranishi, Y., and Shimojo, S.: A lightweight multi-receiver encryption scheme with mutual authentication. In *Proc. of 12th IEEE Intl. COMPSAC Workshop on Security, Trust and Privacy for Software Applications (STPSA 2017)*, pp. 491–497, 2017.

3. Teranishi, Y., Ei Khaing Win, Yoshihisa, T., and Shimojo, S.: A sensor data stream recovery scheme for event-driven IoT applications. In *Proc. of IEEE GLOBECOM 2017*, pp. 1–6, 2017.

3. International Conference Paper (without review)

1. Ei Khaing Win and Mie Mie Su Thwin.: Application of mobile agent and cryptography for the security of job applications on mobile phones. In *Proc. of 3rd Intl. Conf. on Computational Techniques and Artificial Intelligence (ICCTAI 2014)*, pp. 51–55, 2014.

4. Demonstration Paper

1. Ei Khaing Win, Yoshihisa, T., Ishi, Y., Kawakami, T., Teranishi, Y., and Shimojo, S.: A stream merge method to reduce load for sensor data stream delivery. In *Proc. of 4th IEEE Global Conf. on Consumer Electronics (GCCE 2015)*, pp. 405–406, 2015.

5. Domestic Conference Paper

1. Ei Khaing Win, Kawakami, T., Ishi, Y., Yoshihisa, T., Teranishi, Y., and Shimojo, S.: An examination of secure multicast scheme based on user-centric IBE. In *IPSJ SIG Technical Report*, Vol. 2016-DPS-166, No. 31, pp. 1–7, 2016.

2. Ei Khaing Win, Yoshihisa, T., Ishi, Y., Kawakami, T., Teranishi, Y., and Shimojo, S.: A lost sensor data recovery scheme for faster data streams merging. In *IPSJ SIG Technical Report*, Vol. 2017-DPS-169, No. 9, pp. 1–6, 2017.

3. Ei Khaing Win, Yoshihisa, T., Ishi, Y., Kawakami, T., Teranishi, Y., and Shimojo, S.: A data exchange protocol for reliable and secure multicast stream delivery. In *IPSJ SIG Technical Report*, Vol. 2017-DPS-172, No. 5, pp. 1–6, 2017.

Abstract

In the emerging Internet of Things (IoT) technology, the sensors take an essential role for sensing and monitoring. To make IoT applications more responsive and solve some constraints of IoT devices, the sensed data are stored, processed, and analyzed at the edge of the network. For example, in healthcare applications, the body sensor data of a patient may be analyzed on the smartphone of a caregiver instead of cloud to give point-of care efficiently. To deliver the sensed data, reliable and secure sensor data stream delivery is required for IoT applications. As the common sensed data are useful for multiple receivers in many cases, the sensor data stream delivery takes the form of one to many communications. For example, the body sensor data of a patient are delivered to doctor, nurse, and other patients. To reduce the communication overheads caused by the unicast communication for multi-receiver data delivery, IoT technology commonly uses multicast. In multi-receiver data delivery, the receivers encounter data loss for the reasons of unstable communications, insufficient processing powers, and so on. In addition, many IoT applications use sensing data related to personal and sensitive data such as heartbeat data and location data. There have been many studies to tackle problems of data loss recoveries and sensitive data protections. For reliable data processing on IoT applications, the sender needs to generate new recovery streams and deliver them to the loss-encountered receivers. In this way, recovery stream generation causes the processing and communication loads on the sender. In IoT applications, most of the receivers are devices with limited network bandwidth or storage and existing stream merging schemes are not suitable. Regarding sensitive data protections, even though there exist multi-receivers encryption schemes, they do not consider some security properties such as source authentication and replay attack prevention. In addition, multi-receiver encryption schemes must introduce lightweight computational cost, which is important for resource-constrained devices. This thesis proposes a synchronized recovery stream merging scheme and a lightweight multi-receiver encryption scheme to achieve reliable and secure multi-receiver data delivery for IoT. We propose a new recovery stream merging scheme and evaluate its performance.

This thesis consists of five chapters. Chapter 1 explains the research background, research issues, objectives and the content of this research.

Chapter 2 proposes a synchronized recovery stream merging (SRSM) scheme focusing on the data loss issue. Then, it also proposes two methods of the SRSM scheme for IoT applications that analyses the data in order of arrival. The first is latency-aware synchronized recovery stream merging (SRSM-L) method, which is suitable for applications that can tolerate some latency to fetch the lost data such as logging application. The second one is bandwidth dependent synchronized recovery stream merging (SRSM-B) method, which is suitable for IoT applications where the senders have limited network bandwidth. The proposed methods minimize the number of recovery streams and reduce the total bandwidth required by the sender. Through the simulations, we confirm the advantage of our proposed methods compared with existing schemes.

Chapter 3 proposes a certificate-less multi-receiver encryption scheme with authentication for preventing sensitive data leakage from unintended receivers or malicious attackers. The proposed scheme avoids pairing operations that introduce high computational cost. Compared with existing schemes, our proposed scheme shows better results in terms of computational cost and security properties.

Chapter 4 describes the design and implementation of the proposed reliable and secure multi-receiver data delivery for IoT. A synchronized recovery stream merging with a limited number of receivers (SRSM-R) method is proposed to reduce key renewal cost which occurs when a sender changes access policy. To compensate for efficient key renewal cost under the SRSM-R method, the sender has to use larger bandwidth than that of the SRSM-L and SRSM-B methods. Simulation experiments confirmed that the proposed method still achieves better performance than existing schemes.

Finally, Chapter 5 concludes this study and describes the future work of the research.

Contents

Chapter 1

Introduction

1.1 Research Background

Nowadays, low-cost sensors and sensor-attached devices such as smart phones are available. In many applications, sensors are used to detect events or changes in the environment. Various systems that use connected sensors via the Internet are also developed [1, 2, 3, 4, 5]. Due to the sensing and monitoring capabilities, sensors enable the attractive Internet of Things (IoT) technology to provide comfortable life and environment [1] [6, 7, 8, 9]. Many IoT applications have been implemented in various fields such as healthcare monitoring [10, 11, 12], agriculture [13, 14], structural health [15, 16], smart lighting [17, 18, 19], smart parking [20, 21, 22], smart roads [23], smart home [24, 25], and environmental monitoring including air pollution [26, 27], earthquake early detection [28, 29], and so on. On these applications, the sensor data are collected through small devices or smartphones and sent to the cloud for further analysis. On the contrary, the edge computing environment provides attractive model for IoT where many devices at the edge of the network process data immediately and send results to appropriate devices or smartphones. The edge computing can provide more flexible and resilient computational environment for IoT. For example, in [39], Shi et. al. states the edge computing can be applied to the smart home. However, we can not assume large computational nor communication capabilities because edge devices

[1] https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

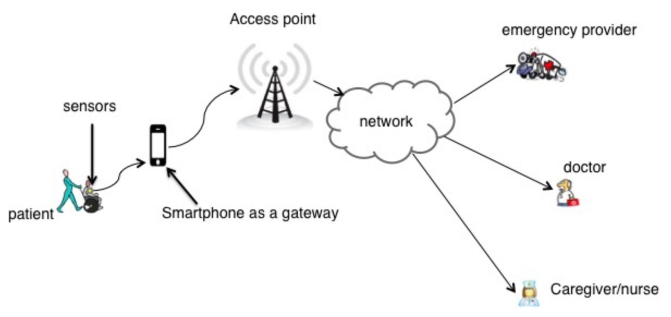


Figure 1.1: An example of a multi-receiver data delivery

such as smartphones or small computers may have limited memory and computing power.

IoT applications on the edge computing environment can be enhanced by delivering sensor data streams to multiple receivers. Data streams are the series of data that are generated continuously. An example of multi-receiver data delivery is shown in Fig. 1.1. The same body sensor data of a patient are useful for healthcare services, family doctors and health monitoring. By delivering body sensor data to all these multiple receivers, the patient can receive these services. In this way, multi-receiver data delivery occur in IoT applications. In such cases, the same data can be delivered effectively using multicast. In this thesis, we focus on the main two issues for multi-receiver data delivery, data loss and security.

Data loss occurs in IoT environment especially for wireless environments. Data loss may arise from different sources such as unreliable network connections, limited power, memory and computational power. For applications that analyze data in order of arrival, data loss is a problem. For example, in the case of patient monitoring, healthcare data are analyzed by using the current body sensor data and its historical ones. In such case, data loss is a problem for providing early warning, preventive care and services. Therefore, data recovery is essential. When more receivers encounter data loss at different timings, the required data sequence differs. As a result, the sender has to prepare different recovery streams. Therefore, reducing the number of recovery streams is a key to reduce bandwidth.

Although the Internet of things (IoT) technology promises benefits to people, it also presents security challenges. In IoT applications that are related to sensitive or personal data, security aspect is very important. The information collected from the sensor might reveal information on individuals, specific locations, and other personal information. For example, the sensed data got from a patient's body are very sensitive. For such IoT applications, the system can avoid data leakages by establishing secure communications. We focus on this issue in this thesis.

In providing reliability and security, energy efficiency is one of the main issues for IoT applications where sensors or sensor-attached devices have limited resources such as limited power capacity, computational resource, and storage. Therefore, energy-efficiency and lightweight schemes are required for these applications.

1.2 Multi-receiver Data Delivery for IoT

In IoT applications, there are many cases where multiple receivers receive the sensor data stream generated continuously by a sensor and utilize it for different purposes. One of the examples is the IoT healthcare application where the smartphone is an IoT gateway of smart things ([91]-[93]) with wireless connectivity. The patient data from body sensors are useful for patient monitoring through streaming to caregivers directly or via a gateway, getting approval for healthcare insurance, getting emergency actions and healthcare guidance, seeking another patient's opinion and estimating the patient's activities. For other examples, the video data streams obtained from surveillance cameras can be used for suspect tracking and congestion detection. The temperature data streams by a roadside sensors can be used for weather analysis and disaster prevention. When the receivers for each system requests data, the sender needs to deliver the same data to multiple receivers. By avoiding one to one communication or unicast communication, IoT applications can introduce a multicast data delivery which can reduce the communication overheads caused by unicast communication.

The IP multicast [30, 31, 32, 42] and application layer multicast (ALM) [33, 34] are commonly used for multicast scheme. In the IP multicast scheme, the receivers join and leave the group by using Internet Group Management Protocol (IGMP) over IP networks. Since the network components such as routers, switches, and cellular network base stations perform the replication of the packet, the sender requires to send a packet only once for a packet delivery to a large number of receivers. However, due to the resource limitations of IoT devices, most of IoT applications using the IP multicast adopt User Datagram Protocol (UDP). UDP is unreliable protocol and does not provide guarantee for a data delivery. Therefore, packets may be lost in UDP. In application layer multicast, the replication is done at the receivers rather than at the network components. Therefore, although the application layer multicast can reduce the loads on network components compared with the IP multicast, it requires high computational load, low data rate and high energy consumption [2].

In this study, we focus on delivering sensitive data such as healthcare data, personal data, and location data. We do not limit on multicast schemes only to the UDP. All we assume for multicast schemes is that the sender can multicast the data streams to multiple receivers. So, Topic based publish/subscribes such as Message Queuing Telemetry Transport (MQTT) [35], Advanced Message Queuing Protocol (AMQP) [3], and Constrained Application Protocol (CoAP) [37] are applicable.

1.3 Research Issues

As described in Section 1.1, there are research issues in multi-receiver data delivery for IoT applications. In this section, we describe two research issues that are tackled in this thesis.

1.3.1 Data Loss Issue

Data loss is one of the important issues in multi-receiver data delivery for IoT applications. Receivers encounter data loss for the following reasons. The first one is the reason caused by loss of network connection. In mobile communication, when the receivers enter into underground or suburbs, they cannot receive electromagnetic waves and lose the network connections. The receivers that lose network connections cannot receive data and lose the data. In IoT environments, different

[2]https://www. cse. wustl. edu/ jain/cse574-14/ftp/coap
[3]http://www.amqp.org

from general problems in UDP, data loss occurs due to the reasons caused by unexpected power on/off of the sender/receiver devices, and some other reasons such as limited device capacities. When the receivers' devices run many softwares, the computation and communication loads become full up, resulting in failure to receive data on their network devices. By ensuring reliabilities, that is recovering the lost data, reliable data analysis can be performed. One of the basic mechanisms to ensure the reliabilities is data retransmission. In this mechanism, the sender has to retransmit the lost data to achieve reliable data delivery. Because the sender has to generate new recovery streams consisting of the lost data for all receivers who request the lost data, retransmission leads to overhead in bandwidth on the sender. Therefore, the sender's load is directly proportional to the number of recovery streams in solving the data loss problem.

A large number of existing works have focused on stream merging schemes to reduce the number of recovery streams [48, 49, 53, 54, 55]. However, these works mainly focus on video-on-demand applications where the jitter or the data delivery interval needs to be small to provide smooth video playback. The jitter then becomes the critical factor that limits the number of recovery streams that can be merged with other recovery streams. To compensate for the jitter, in some existing works [53, 54, 55], the receivers have to receive more than one data stream at the same time. To merge the streams faster, some existing works apply the receive-two models that allows the receivers to receive two streams at the same time; one for viewing purpose and another one for future use [53, 54, 55]. Therefore, the receivers need to have a large buffer storage. However, a large storage cannot be expected in IoT devices. In other words, the receivers may not have an enough buffer storage. Therefore, the existing works are inappropriate for IoT environments.

1.3.2 Data Leakage and Security Issues

Another issue in multi-receiver data delivery is data leakage. The leakage of IoT sensor data to unauthorized receivers is the biggest risk when the data are personal or sensitive. For example, in healthcare application, the information gathered from the wearable devices could be used to physically attack the person. To obtain the unauthorized data access or change the data in some way, the attackers will make several attempts. The attackers will try to discover the sensitive data, modify the data in transit, impersonate as the valid sender or receiver, and replay the valid data transmission. To prevent sensitive data leakage and solve the security issues, security properties such as confidentiality, message authentication, and replay attack prevention are very important. Therefore, security is the basic requirement in the provision of IoT applications.

To preserve confidentiality and message integrity, several certificate-based and certificate-less multi-receiver encryption schemes have been proposed [78, 83, 84, 85]. However, these schemes do not avoid computation expensive operations and some of these schemes lack some security properties such as replay attack prevention, source authentication, and implicit user authentication. Moreover, IoT devices do not have enough storage and computing power. So, heavy computational load is not a desirable property for IoT environments consisting of highly constrained interconnected devices. NIST provides good survey about lightweight cryptography for resource constrained devices. In [61], NIST-approved cryptographic standards for authenticated encryption such as Counter with Cipher Block Chaining-Message Authentication (CCM) and Galois/Counter Mode (GCM) block cipher modes are introduced. However, CCM is not designed for stream processing. Both CCM and GCM are modes of operation of the symmetric key block cipher. GCM is a mode of operation of the AES algorithm and it can provide only data confidentiality and data authenticity. Since there is scalability problem in symmetric key algorithms, they are suitable for small scale IoT applications. Lightweight cryptography standards also include elliptic curve based authentication scheme. The elliptic curve cryptography is the optimal solution for real time embedded systems in IoT networks [62]. In large IoT applications such as healthcare application, poor scalability is not suitable for applications. Moreover, source authentication and replay attack prevention are important. Therefore, the existing works are not enough. In this thesis, we will use elliptic curve cryptography to achieve necessary security requirements.

1.4 Research Objectives

The goal of this thesis is to solve the problems caused by the above two issues in multi-receiver data delivery for IoT. The research objectives are as follow.

1. Recovery Data Stream Reduction for Reliable Data Delivery
We assume that the receivers who encountered data loss request the recovery streams. the number of recovery streams is proportional to the number of recovery requests. Generally, the number of recovery requests increases when the number of receivers increases and thus, the sender's load for delivering the recovery data streams increases. A larger number of recovery data streams causes more loads. Therefore, the research objective for reliable data delivery is reduction of the number of recovery streams.

2. Lightweight Computations for Secure Data Delivery
For secure data delivery, encryption takes an important role. Depending on the number of receivers, the sender's load for encryption differs. The sender's load for encryption increases as the number of receivers increases. For multi-receiver data delivery, encryption time increases. In order to be adaptable for IoT environment, we have to propose a lightweight and secure multi-receiver encryption scheme. In the proposed scheme, computation expensive operations should be reduced to achieve lightweight computational cost. And the necessary security requirements such as confidentiality, message integrity, source authentication, user authentication, and replay attack prevention should be fulfilled. So, the research objective focusing on this thesis is a lightweight multi-receiver encryption scheme for secure data delivery.

3. Key Renewal Cost Reduction for Reliable and Secure Multi-receiver Data Delivery
To avoid data leakage, the sender needs to renew the key whenever any one of the receivers leaves. This is because the disjoined receiver knows the old key. The sender has to deliver the renewed key to the remaining receivers in secure ways. The key renewal cost increases when the number of receivers increases. In the case of high key renewal cost, the sender needs to use more CPU power. So

the research objective for reliable and secure multi-receiver data delivery is key renewal cost reduction.

Each of these three objectives corresponds to each topic of the chapters in the thesis.

1.5 Organization of Thesis

This thesis consists of five chapters, and the rest of this thesis is organized as follows.

In the next section, we introduce a synchronized recovery stream merging (SRSM) scheme and describe the previous works related to the recovery streams reduction. Then, an assumed system model for data stream delivery is explained in detail. In our proposed scheme, we adopt two methods of synchronized recovery stream merging scheme, latency-aware synchronized recovery stream merging (SRSM-L) method and bandwidth-dependent synchronized recovery stream merging (SRSM-B) method. In the SRSM-L method, the sender merges the streams according to the receivers' constraints such as acceptable latency and skip rate. In the SRSM-B method, the sender has limited bandwidth for data delivery. Depending on the available bandwidth of the sender, the SRSM-B method merges the recovery streams. We also conduct the simulation experiments to verify that our proposed methods reduce the number of recovery streams.

In Chapter 3, we explain the security issues and propose a certificate-less multi-receiver encryption scheme with authentication (CLAME) based on elliptic curve cryptography. We firstly introduce the preliminaries of our proposed scheme including the definition of elliptic curve cryptography, the framework of the proposed scheme, chosen ciphertext attacks and security model. Then, the details of the proposed scheme are discussed. In addition to experimental results, security proofs are also shown in this chapter.

In Chapter 4, we propose the design and implementation of reliable and secure multi-receiver data delivery for IoT. Firstly, we discuss the related work and issue in this topic. Then, we explain the proposed system model and a method called the synchronized recovery stream merging with a limited number of receivers

(SRSM-R) method. The SRSM-R method is proposed to reduce the load of shared secret renewal time cost. For secure data delivery, data is encrypted by using certificate-less multi-receiver encryption (CLAME) proposed in Chapter 3. Our proposed method has advantages over other methods in terms of the key renewal time cost. The experiments are conducted to examine the benefits of the proposed method.

Finally, we summarize our work and present future works in Chapter 5.

Chapter 2

A Lost Data Recovery Scheme for Sensor Data Stream Multicasting

2.1 Introduction

As described in Chapter 1, in some IoT applications, multiple receivers receive sensor data stream generated continuously by sensors and utilize it for various objectives. For example, a sensor data stream obtained by the body sensors of a patient can be used for patient monitoring, health insurance approval, healthcare data sharing, and patient activities estimation, and so on. In the edge computing environment [38], where many computers run on the network edges to realize quick-response on IoT applications, a large number of edge computer devices must receive the sensor data stream. To deliver the data stream to multiple receivers efficiently, we assume that the network has an ability of multicast, such as the ad-hoc sensor networks [40, 41], IP multicast [42], and application layer multicast like pub/sub messaging [43]. The pub/sub messaging is already widely utilized by many IoT applications [44]. Some existing works implement publish/subscribe messaging on distributed application layer multicast [45, 46].

In IoT applications, the data losses can occur because of the unstable network, unexpected power on/off of the sender/receiver devices, electricity black-offs, and some other reasons. In healthcare data analysis, the data must be clean and lossless to improve care and save life. In other words, reliable data is important.

Typically, the sender waits for recovery request and retransmits the lost data to each loss-encountered receiver. Since the data loss occurs at different timings for different receivers, the number of recovery streams grows and consumes network bandwidth. In the edge computing environment, the sender does not have plenty of CPU power nor network ability. That means the network of the sender is limited in bandwidth. Therefore, it is important to reduce the number of streams for recovery to ease the network bandwidth.

To reduce the number of streams and alleviate network bandwidth usage on the sender, stream merging schemes ([48]-[59]) are proposed. *Stream merging* means merging the recovery streams occurred at the close timing. Obviously, stream merging reduces the number of streams delivered by the sender and reduces network bandwidth usage. Because these existing schemes assume mainly Video-on-Demand (VoD) applications, they need to keep the jitter of data stream delivery as small as possible for smooth video playback. However, the jitter is not critical in IoT applications that analyze the data in order of arrival. In addition, to obtain the real-time information of the real world, the arrival order of data is important because the analysis process has its own context, which means the transitions of the status in the analysis are decided based on the previous status. For example, in the case of patient monitoring, healthcare data analysis is performed by using the current body sensor data and its historical ones. In the case of checking pulse, if the sensed data from heartbeat sensor are lost for 10 sec, the irregular pulse can be missed. The body sensor data are sent through data streaming to caregivers. In such case, data loss is a problem for providing care and services. If the receiver has a plenty of buffers to store received data, he or she can adjust the order of the data according to the timestamp of the data even when the arrival order of data is not preserved. Besides, most of the existing schemes assume that there is a large size buffer on the receiver. We cannot expect such a large size buffer on IoT applications that are often deployed as embedded systems in edge computing environment. In IoT healthcare application, caregivers use smartphones to provide point-of-care. Since a caregiver has to handle multiple patients at the same time and the device may encounter unexpected power on/off, recovery stream delivery needs to consider a solution for efficient scheduling of available resources such as buffer size of the caregiver's device.

In this chapter, we propose a new multicast delivery scheme called synchronized recovery stream merging (SRSM) scheme focusing on the data loss issue in IoT. To avoid data loss issue while considering the limitations of data streaming among resource-constrained devices, we propose two synchronized recovery stream merging methods which keep the data of arrival for IoT applications as much as possible. The first one is “Latency-aware synchronized recovery stream merging (SRSM-L)” method for IoT applications where the receivers can tolerate the sender’s data delivery latency as long as the delivery latency is in the range of their acceptable latencies, and the other is “Bandwidth-dependent synchronized recovery stream merging (SRSM-B)” method for the cases where the network bandwidth for the sender is limited. According to the simulation results, we also confirm that our proposed sensor data recovery methods can reduce the number of streams in the different failure scenarios.

The remainder of this chapter is organized as follows. In Section 2.2, we introduce the related work. Section 2.3 describes the basic description in the stream merging scheme. In Section 2.4, the synchronized recovery stream merging, the two methods of the proposed scheme and the detailed algorithms of the two methods are explained in detail. We show the evaluation by simulations in Section 2.5. Finally, we conclude this chapter in Section 2.6.

2.2 Related Work

To reduce the number of streams, some data stream merging schemes have been proposed such as Batching [48], Piggybacking [49], Tapping [53], Patching [54], and Dynamic skyscraper [55]. These are originally proposed for Video-on-Demand applications.

In the Batching technique, the sender’s bandwidth required for delivering video data is reduced by delaying the start of the delivery. The senders define a specific time interval and deliver the stream at the end of each time interval. However, the Batching introduces delay on the receivers. To merge the streams faster, the speeds of the stream deliveries are dynamically adjusted in the Piggybacking. In the Piggybacking, the receivers do not need to have large buffer storage. In the Tapping and the Patching, the receivers receive the stream data delivered for other receivers and store the data in a short time until they use the data. In the Dynamic skyscraper, the sender's bandwidth is reduced by allowing the receivers to receive more than one stream at the same time.

In [58], efficient dyadic stream merging algorithm that allows the receivers to receive up to two streams at any time was proposed. The Dyadic Tree is constructed by assuming the original stream as a root (parent) and the earliest receiver who arrived within a designated interval as the children. The streams for the children are merged with the stream for the parent. Moreover, the streams for the receivers who arrived later and the receivers who arrived within the same time interval are merged again. In this way, the number of streams that the sender delivers is reduced. However, the receivers need to have the buffers to simultaneously accept two streams at any time.

Bar-Noy et. al compare five stream merging algorithms for media-on-demand in [59]. Considering distributions for the request patterns of the receivers, they present empirical study results for the dynamic Fibonacci, dyadic and earliest reachable merge target (ERMT). Simulation results are shown for various performance metrics. Most of them use the buffer on the receiver.

Because the application targets of the above existing schemes are VoD applications, they assume that the data delivery intervals are almost the same. In other words, the jitters of data delivery interval are small so that a person cannot notice the jitters for the smooth video playback. By this assumption, the number of streams that the sender can reduce is limited. However, the jitter is not critical in IoT applications. In this chapter, we assume that the receivers do not have such large buffer to store data because IoT applications can run on small devices such as smartphones. Therefore, the receivers are assumed to receive only one stream at a time. In this chapter, we propose a new stream merging method assuming that the device has no large buffer and can receive single stream at a time.

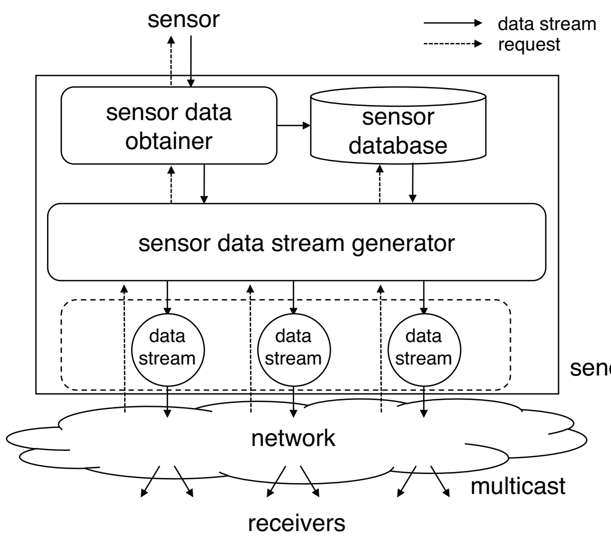


Figure 2.1: Assumed System Model

2.3 Basic Description in Stream Merging Scheme

Fig. 2.1 shows our assumed system model. The data stream sender consists of sensor data obtainer, sensor database, and sensor data stream generator. We assume that the sender components typically run on the gateways of the sensor network, which have larger computation and storage capabilities.

2.3.1 Principle Components of the Data Stream Sender

The sensor data obtainer obtains the sensed data from the sensors periodically and continuously. All obtained data are stored into the sensor database. Normally, the receivers request the sensor data stream that delivers sensor data immediately after the data is obtained by the sensor data obtainer. Hereafter, we call it as the *original stream*. The request is issued only once from the network, because we assume that the data stream is delivered by multicast. The receivers of the original stream subscribe to the multicast group which the original stream corresponds to. Once the request for the original stream is accepted, the sensor data stream generator generates an original stream and starts delivery to the network.

2.3.2 Skipped Data Delivery

When a receiver encountered a failure, which means a receiver encountered a data loss, a new data stream request is issued so that the loss-encountered receiver can obtain data which are delivered by the original stream during the failure. The new data stream requested by the loss-encountered receiver is called the *recovery stream*. As a response to the request, the sender sends the identifier that corresponds to the recovery stream. The receiver joins to the multicast group which corresponds to the identifier. The recovery stream can be either a larger bandwidth data stream, which includes more data per unit time than the original stream or a skipped data stream, which drops some data from the original stream. The skipped data stream is only applicable when the application allows a low temporal resolution. For example, the location log of a patient is important in estimating the patient's activities. However, the location logging consumes heavy battery power. To save the battery life, an option is location logging with lower logging rate. And this case is similar to skipping sensor data such as skipping location log for some time. Therefore, in the proposed scheme, skipping alternate frames or some amount of frames is allowed. Unlike other frame skipping methods, the proposed scheme does not consider the skipping scheme. In simplicity, the proposed scheme skips one frame after choosing one frame for retransmission if the skip rate is 2.0.

In the following, we define a unit time is equal to the time needed to deliver single unit data in the original transfer rate. Bandwidth that is required for sending an original unit data in the unit time is defined as 1. The sender delivers the recovery stream according to the receiver's request. For the recovery stream with skipped data, we define a *skip rate* that determines how much data are skipped within a unit time. The number of data units skipped within a unit time is equal to (skip rate-1)$\times$(original data units within a unit time). The original stream's skip rate is 1.0. For recovery streams, defined skip rate value must be equal to or greater than 1.0. The bandwidth is the data amount that the sender sends within a unit time. For example, if the skip rate is 2.0 and bandwidth is 1.0, the sender skips one data within a unit time.

In Fig. 2.2, we assume that the sender delivers the original data stream with

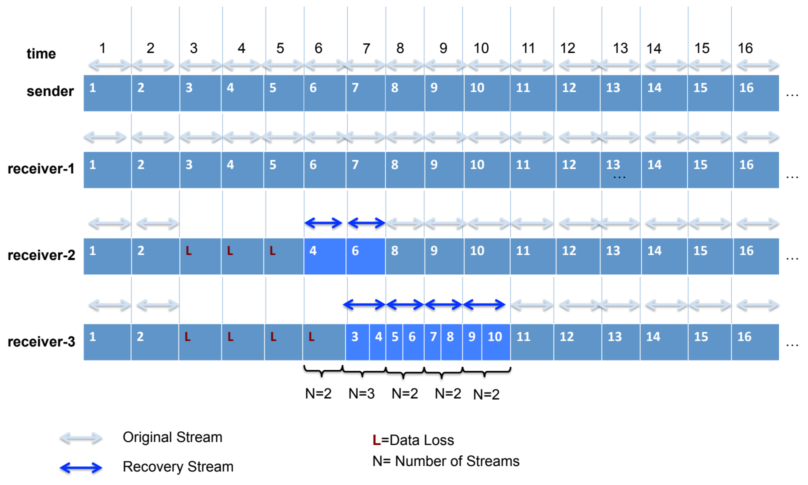


Figure 2.2: An example of skipped data delivery

bandwidth=1.0. Fig. 2.2 shows the situation in which the receiver-2 sets skip rate as 2.0 and bandwidth as 1.0 while the receiver-3 sets skip rate as 1.0 and bandwidth as 2.0 for lost sensor data recovery where data transfer rate is double. As the receiver-2 sets skip rate as 2.0, the sender skips one data before sending one recovery data. The quantity of data the sender eliminates from the recovery stream is a parameter and can be adjusted depending on the data importance level specified by the requested receiver. As the eliminated data increases, the recovery stream can faster catch up with the original stream. In Fig. 2.2, at time 6, the sender uses total bandwidth 2.0 for the original stream and the recovery stream. At time 7, the sender has to use total bandwidth 4.0 as the number of recovery stream increases. Therefore, the bandwidth of the sender is proportional to the number of recovery streams. If there are multiple recovery requests for different failure situations, the sender may exhaust maximum bandwidth. Therefore, not only the receiver but also the sender should be considered in the lost data recovery scheme.

2.3.3 Basic Behavior of the Recovery Stream Delivery

Fig. 2.3 illustrates the basic behavior of the recovery stream delivery. The x-axis represents the elapsed time and y-axis corresponds to the delivered data amount, that is the delivered sequence number. In Fig. 2.3, a receiver starts to recover

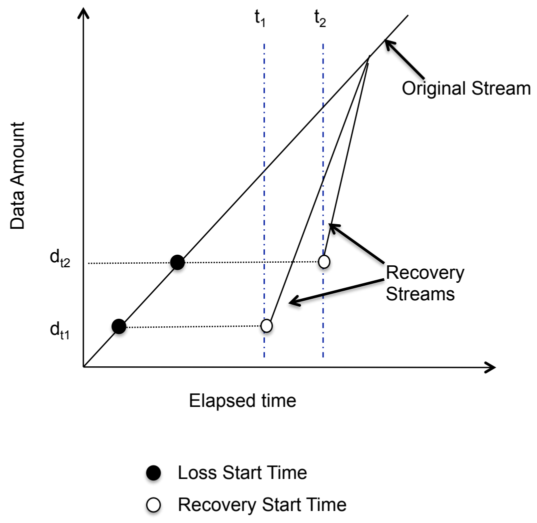


Figure 2.3: Recovery Stream Delivery

from the failure at time (t_1) and the sender needs to generate a new stream by using the receiver's desired skip rate if there is no existing recovery stream. When another receiver with different failure condition requests for recovery at time (t_2), the sender has to set a new recovery stream. This time, the sender chooses faster recovery rate. A recovery stream with larger skip rate or larger bandwidth faster catches up with the original stream.

Although the recovery stream can faster catch up with the original stream, the number of received data on the receiver may decrease. When the recovery streams catch up with the original stream, the sender can deliver the same original stream to the receivers that have already received the lost sensor data. Therefore, the sender does not need to generate the recovery stream and the number of streams is reduced.

The critical problem in such data delivery is how to alleviate the network bandwidth usage of the sender. When more loss-encountered receivers with different failure situations are involved in recovery, the sender's load such as the computational power, memory usage, communication traffic, etc. increases. In other words, the sender's load is proportional to the number of recovery streams.

2.4 Proposed Scheme

In this section, we propose a new stream recovery scheme called the synchronized recovery stream merging (SRSM) firstly. Then, as the extensions of the proposed scheme, two stream data recovery methods called "Latency-aware synchronized recovery stream merging (SRSM-L)" and "Bandwidth-dependent synchronized recovery stream merging (SRSM-B)" are explained in detail.

2.4.1 Synchronized Recovery Stream Merging (SRSM)

In our proposed SRSM scheme, we assume that each receiver has a time restriction, which we call the *acceptable latency*. The acceptable latency means tolerable latency from the time when the original data was generated. The acceptable latency is specified according to IoT application demands. Example scenarios are:

- For getting healthcare insurance approval, the past data of a patient is important. Therefore, logging healthcare data is important. Before healthcare insurance application does not happen, logging application can tolerate latency. In this case, the receiver may wait lost data for a half day if the insurance application will be done after two days. Therefore, the acceptable latency is equal to a half day.

- For getting healthcare advice, real-time data analysis is preferable to improve care or save life. In this case, lower acceptable latency value is a requirement.

The acceptable latency may depend on the receivers. Different receivers may have different latency tolerances and different specifications of data importance on the same data. Using the nature of different applications, the proposed scheme tries to save the network bandwidth usage of the data sender.

In our proposed scheme, stream merging occurs when the recovery stream catches up with the original stream and it is also performed among recovery streams. Recovery streams are merged when they had received the same data amount. The notations that are used as the parameters of the receiver-*i* for the recovery are shown in Table 2.1.

Table 2.1: Recovery parameters of the receiver-i

Parameter	Description
r_i	the acceptable skip rate of the receiver- i
b_i	the acceptable bandwidth of the receiver-i
t_i	the recovery time of the receiver- i
A_i	the total data amount the receiver-i possesses before t_i
d_i	the last time of the receiver-i that received A_i
D_i	the total data amount the receiver-i will possess after t_i
l_i	the acceptable latency of the receiver-i

Table 2.2: Parameters of the recovery stream

Parameter	Description
R_s	the skip rate of the recovery stream s
B_s	the bandwidth of the recovery stream s
A_s	the total data amount that s has delivered before t_i
d_s	the last time of s that delivered A_s
D_s	the total data amount that s will possess after t_i
G_s	the set of members of the multicast group in s

In the SRSM scheme, the sender does not send some lost sensor data as described in subsection 2.3.2. In this way, the data amount received by the recovery stream quickly catches up with that delivered by the original stream. If there is no existing recovery stream when the receiver-i recovered from the failure, the proposed scheme creates new recovery stream (s) with the parameters described in Table 2.2.

Then, it adds the receiver-i into G_s. Any receivers whose received data amount is equal to that of the group of receivers of s are added as a group member without creating a new recovery stream. Otherwise, whether the receiver-i can be added to the group or not is determined according to the recovery parameters described in Table 2.1, expected catchup time, and related latency.

If the amount of data already received by the receiver-i is more than that of the existing recovery stream (s), the expected catch up time of stream (s) to the receiver-i ($C_{s,i}$) and the related latency ($L_{i,s}$) can be calculated as follows:

$$\begin{aligned}
C_{s,i} &= \frac{(D_i - D_s)}{R_s \times B_s} + t_i && \text{if } A_i > A_s \\
L_{i,s} &= C_{s,i} - (d_i + 1) && \text{if } A_i > A_s
\end{aligned}$$

where

$$\begin{aligned}
D_i &= A_i + r_i \times b_i \\
D_s &= A_s + R_s \times B_s
\end{aligned}$$

The related latency ($L_{i,s}$) is the latency of the receiver-i to merge into the existing recovery stream (s). Although the receiver-i's received data amount exceeds that of the existing recovery stream (s), the receiver-i has to wait until the existing recovery stream has delivered the same data amount that it has already received. In other words, the related latency is the waiting time of the receiver-i to get the data of the existing recovery stream.

When the data amount already delivered by the existing recovery stream (s) exceeds that of the receiver-i, the calculations of the expected catch up time for the receiver-i and related latency of the existing recovery stream ($L_{s,i}$) are as follows:

$$\begin{aligned}
C_{i,s} &= \frac{(D_s - D_i)}{r_i \times b_i} + t_i && \text{if } A_s > A_i \\
L_{s,i} &= C_{i,s} - (d_s + 1) && \text{if } A_s > A_i
\end{aligned}$$

For stream merging among the receivers who received different data amount, we need to consider two cases.

The first case is to continue delivering the existing recovery stream (s), which exists before new recovery request, and the sender adds new recovery requested receiver (receiver-i) into the group of s. Two requirements exist for the first case. The first requirement is that the data amount delivered by s must be less than that of the receiver-i. Since the sender adds the receiver-i into the group of s, the receiver-i has to wait until the receivers of s receive the same data amount of the receiver-i. Therefore, the second requirement is that the related latency must be less than or equal to the acceptable latency of the receiver-i. In summary, the requirements for the first case can be described as follows:

$$L_{i,s} \leq l_i \qquad \text{when } A_i > A_s$$

And the second case is to create a new recovery stream for the new recovery requested receiver (receiver-i). The sender drops the existing recovery stream (s) and adds the receivers of s into the group of new recovery stream. In the second case, the data amount received by the receiver-i must be less than that of s. Since the receivers of s are added to a new recovery stream, the minimum acceptable latency of the receivers of s needs to be considered. If the related latency is less than or equal to the minimum acceptable latency of the receivers of s, stream merging is performed. The requirements for this case can be described as follows:

$$L_{s,i} \leq L_{min:s} \qquad \text{when } A_s > A_i$$

$L_{min:s}$ is the minimum acceptable latency among the receivers of the existing recovery stream (s).

In both cases, the sender decides to start delivery of the data stream for the receivers who received smaller data amount so that other receivers will not lose any more data. For example, at time $t1$, the sender delivers one recovery stream. Suppose that the recovery stream delivered at time $t1$ is an existing recovery stream (s). At time $t2$ where ($t2 > t1$), there is another recovery request. At time $t2$, the existing recovery stream has received 60 data frames. However, the receiver who requests the recovery has received only 40 data frames. In this scenario, the receiver who requests the recovery will lose 20 data frames if the sender chooses the first case stream merging without considering the requirements. The scenario meets the first requirement of the second case merging. If the sender decides to create a recovery stream for a new recovery request and drops s, the receivers of s have to wait until the new recovery stream's receivers received 60 data frames. If the second requirement is fulfilled, any receivers will not lose more data.

In summary, if there is one or more recovery streams, the proposed scheme merges the recovery streams for the following three cases.

1. Case-I: A merge-able stream exists but not reached to D_i

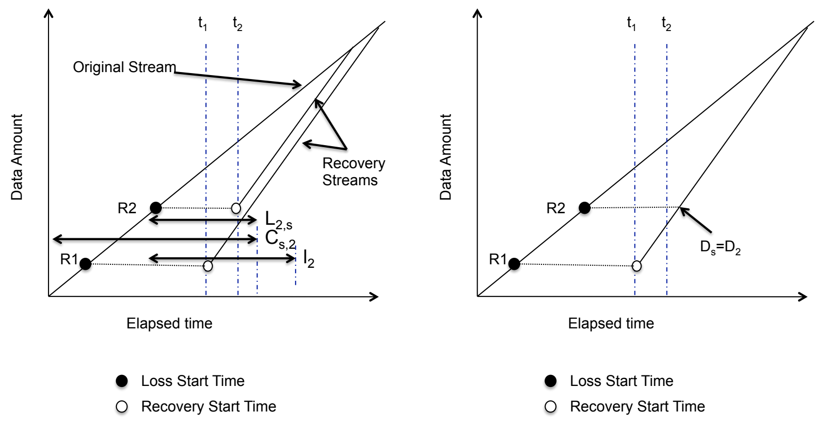


Figure 2.4: Case I: Before merging vs After merging

In the first case, the proposed scheme adds the receiver-*i* recovered from the failure to the existing recovery stream (s) if the following conditions are satisfied.

$$scontains(R_s, r_i, B_s, b_i) \tag{2.1}$$

$$D_s < D_i \tag{2.2}$$

$$C_{s,i} - (d_i + 1) \leq l_i \tag{2.3}$$

The function *scontains*() checks whether the skip rate and the bandwidth of the existing recovery stream s is acceptable for the receiver-*i*. The second statement means the existing recovery stream has not received the data d_i already received by the receiver-*i*. Eq. (2.3) checks whether the related latency satisfies the acceptable latency of the receiver-*i*.

An example of Case I stream merging is shown in Fig. 2.4. In Fig. 2.4, we denote R1 and R2 as two receivers. Starting at (t_1), there is one existing recovery stream (s) for the receiver-1. At the recovery request time (t_2) of the receiver-2, the recovery data count status is ($D_2 > D_s$). We assume that the receiver-2 can accept the skip rate and bandwidth of the existing recovery stream (s) and tolerate the waiting time to get recovery data from stream (s). In other words, the conditions meet the requirements for Case I merging. Therefore, the receiver-2 is added to the existing recovery stream without creating new recovery stream.

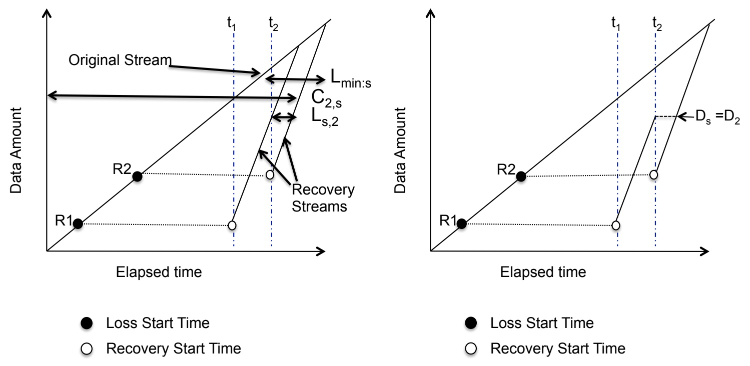


Figure 2.5: Case II: Before merging vs. After merging

2. Case II: A merge-able stream exists and already delivered D_i

Whether the new recovery stream should be generated is checked using the following three conditions.

$$scontains(R_s, r_i, B_s, b_i) \tag{2.4}$$

$$D_s > D_i \tag{2.5}$$

$$C_{i,s} - (d_s + 1) \leq L_{min:s} \tag{2.6}$$

The first condition checks whether the skip rate and bandwidth usage of the receiver-i is acceptable for the existing recovery stream (s). The second condition means that the existing recovery stream has already delivered more data than that required by the receiver-i. The final condition checks whether the related latency satisfies the acceptable latencies of all receivers in the existing recovery stream so that all receivers of the existing recovery stream can wait until new recovery stream delivers D_s.

Fig. 2.5 shows the second situation in which the recovery requested by the receiver-2 has less recovery start data count than that of the existing recovery stream. When the receiver-2 starts its recovery at t_2, there is already one existing recovery stream (s) for the receiver-1 and recovery data count status is ($D_s > D_2$). We assume that the first conditional statement and the final conditional statement for latency are also satisfied. Therefore, the receivers in the existing recovery stream are added as the receivers of new recovery stream before eliminating the existing recovery stream. Finally, there is only one recovery stream.

3. Case III: A coincidental merge-able stream exists

When the receiver and the existing recovery stream (s) have the same recovery data to be delivered, and the skip rate and bandwidth usage of s are acceptable for the receiver-i, the receiver is added to the group of the existing recovery stream. The following conditional statements check whether the coincidental merge-able stream exists.

$$scontains(R_s, r_i, B_s, b_i) \tag{2.7}$$

$$D_s = D_i \tag{2.8}$$

2.4.2 Latency-aware Synchronized Recovery Stream Merging (SRSM-L)

We propose the "Latency-aware synchronized recovery stream merging (SRSM-L)" method on the basis of the SRSM scheme. We assume the latency-aware environment for the SRSM-L method in which the receivers specify the acceptable latency to merge the streams. In the SRSM-L method, the sender does not create a new stream as long as the existing recovery stream can satisfy the receiver's constraints. Once the sender receives the request, the sender decides whether to generate a new recovery stream or add the requested receiver as a receiver of one of the existing recovery streams.

The algorithm for the SRSM-L method is shown in Algorithm 1 where *streams* represents the streams to be delivered at time t and the latency of each receiver-i of *streams* does not exceed corresponding acceptable latency.

Firstly, the proposed scheme checks whether there is coincidental merge-able stream. If the recovery requested receiver-i and one of the existing streams have the same data recovery start data amount and the skip rate and bandwidth are acceptable for the receiver-i, the receiver is added to the group of the existing recovery stream. Among the acceptable latencies of all receivers who are members of the existing recovery stream, the minimum acceptable latency is set as the acceptable latency of the existing recovery stream. Conditional statements from line

Algorithm 1: Algorithm for the SRSM-L method

```
procedure SUBSCRIBE
Input: $t$, $i$, $r_i$, $b_i$, $d_i$, $l_i$
Initialize empty stream array $z$
$D_i \leftarrow A_i + r_i \times b_i$
foreach $s \in streams$ do
    $D_s \leftarrow A_s + R_s \times B_s$
    if $(D_s = D_i)$ and $(scontains(R_s, r_i, B_s, b_i))$ then
        $l_i \leftarrow l_i - (t - (d_i + 1))$
        $w_i \leftarrow (t - (d_i + 1))$
        $s$.addReceiver($i$)
        $L_s \leftarrow L_{min:s}$
        return
    end
    if $(D_s < D_i)$ and $(L_{i,s} \leq l_i)$ and $(scontains(R_s, r_i, B_s, b_i))$ then
        $l_i \leftarrow l_i - (C_{s,i} - (d_i + 1))$
        $w_i \leftarrow (C_{s,i} - (d_i + 1))$
        $s$.addReceiver($i$)
        $L_s \leftarrow L_{min:s}$
        return
    end
    if $(R_s != 1.0)$ and $(D_s > D_i)$ and $(L_{s,i} \leq L_s)$ and
     $(scontains(R_s, r_i, B_s, b_i))$ then
        $z$.addStream($s$)
    end
end
Create new stream $ns$
$l_i \leftarrow l_i - (t - (d_i + 1))$
$w_i \leftarrow (t - (d_i + 1))$
$ns$.addReceiver($i$)
foreach $s$ in $z$ do
    foreach $u$ in $s.receivers$ do
        $w_u \leftarrow (C_{i,s} - (d_s + 1))$
        $l_u \leftarrow l_u - (C_{i,s} - (d_s + 1))$
        $ns$.addReceiver($u$)
    end
    $z$.deleteStream($s$)
end
$L_{ns} \leftarrow L_{min:ns}$
$streams$.addStream($ns$)
end procedure
```

number (7) to line number (12) check the situation for Case III merging. If the condition is true, stream merging is performed.

Although the receiver and the recovery stream have different recovery start data amount, stream merging may occur depending on the acceptable latency of the receiver-i or the recovery stream. The proposed method always delivers the recovery stream with the smaller data amount. If the existing recovery stream has delivered smaller recovery start data amount than that required by the recovery requested receiver-i, the proposed method continues it without generating new recovery stream for the recovery requested receiver-i. The acceptable latency of the receiver-i with larger data amount is considered because it has to wait until the existing recovery stream (s) delivers its recovery start data amount. Therefore, the related latency ($L_{i,s}$) must be less than or equal to the acceptable latency (l_i) so that the receiver-i can be added to the existing recovery stream (s). The acceptable latency (l_i) of recovery requested receiver-i is updated for time (t) so that the sender considers the remaining latency of the receiver-i in merging with other streams or other receivers at the same time (t). Therefore, the acceptable latency of the recovery stream (L_s) is modified with the minimum acceptable latency among the receivers of the recovery stream ($L_{min:s}$) whenever a receiver is added to the existing recovery stream. Moreover, the latency (w_i) of receiver-i is needed to be updated to find its maximum latency throughout its processing time. Conditional statements from line number (13) to line number (18) are grouping receivers for the existing recovery stream. These statements are the requirements for Case I merging. The summary of this case can be described like this:

$$G_s = G_s \cup \{i\} \text{ when } L_{i,s} \leq l_i \text{ and } A_i > A_s$$

In case of the existing recovery stream with more recovery start data count ($D_s > D_i$), the proposed scheme generates new recovery stream for the recovery requested receiver and eliminate the existing recovery stream. The acceptable latency of the receivers from the existing recovery stream needs to be considered. In checking the acceptable latency, the related latency ($L_{s,i}$) must be less than or equal to the acceptable latency (L_s) so that new recovery stream will be generated by dropping the existing recovery stream. Conditional statements starting from line number (19) to line number (20) describe the above situation. In other words, the statements are the conditional statements for Case II merging. The summary of this case is

$$\begin{aligned}&G_{ns} = G_s \cup \{i\} \text{ when } L_{s,i} \leq L_s \text{ and } A_s > A_i\\ &\text{delete } G_s\end{aligned}$$

where G_{ns} is the group of receivers in new recovery stream.

The receiver-i and the receivers of the existing recovery stream are added to new recovery stream (ns) before dropping the existing recovery stream. Then, the existing recovery stream is removed and the acceptable latency of new recovery stream (L_{ns}) is assigned with the minimum acceptable latency among its receivers ($L_{min:ns}$). The conditional statements described from line number (21) to line number (32) correspond to new stream generation and latency assignment of new stream and its receivers.

2.4.3 Bandwidth-dependent Synchronized Recovery Stream Merging (SRSM-B)

We also propose the "Bandwidth-dependent synchronized recovery stream merging (SRSM-B)" method on the basis of the SRSM scheme. In the real scenario, the available bandwidth of the sender may dynamically change time by time. Instead of satisfying the receiver requirements, the SRSM-B method prioritizes the sender's one.

In the SRSM-B method, the sender considers the receivers' latencies as much as possible in merging streams while considering its available bandwidth. Firstly, it performs streams merging by considering the receivers' acceptable latencies. Then, it checks the required bandwidth and the sender's available bandwidth. In case of the bandwidth requirement exceeds the available one, it again merges some streams until the necessary bandwidth fits into the sender's available bandwidth. As a result, some the receivers may exceed the acceptable latencies. However, SRSM-B merges the streams with the least latency difference to make excess latency value as small as possible. The sender needs to have the following parameter

Algorithm 2: Algorithm for the SRSM-B method

```
1  procedure MERGE
2  Performs SUBSCRIBE procedure of SRSM-L for all recovery requested
     receivers at time $t$
3  if $(n_t \leq a_t)$ then
4      break
   end
5  else
6      $s_1 \leftarrow stream_{t(ori)}$
7      $s_2 \leftarrow stream_{t(ori)}$
8      $s_t$.deleteStream($s_2$)
9      $q = n_t - 1$
10     $m = \frac{q}{a_t - 1}$
11     foreach $(i = 1$ to $q)$ do
12         $s_3 \leftarrow stream_{t(min)}$
13         foreach $s$ in $mergestream$ do
14             if $(s_{count} < m)$ then
15                 add all receivers of $s_3$ to $s$
16                 if $(R_s > R_{s_3})$ then
17                     $R_s = R_{s_3}$
                   end
18                 if $(B_s > B_{s_3})$ then
19                     $B_s = B_{s_3}$
                   end
20                 foreach $u$ in $s_3.receivers$ do
21                     $w_u \leftarrow (C_{s,s_3} - (d_{s_3} + 1))$
22                     $l_u \leftarrow l_u - (C_{s,s_3} - (d_{s_3} + 1))$
                   end
23                 $s_t$.deleteStream($s_3$)
24                 increment $s_{count}$ by 1
25                 return
               end
           end
26         $mergestream$.addStream($s_3$)
27         $s_{3count} = 1$
28         $s_t$.deleteStream($s_3$)
29         increment $i$ by 1
       end
30     $mergestream$.addStream($s_1$)
   end
31 end procedure
```

values for bandwidth-dependent recovery stream delivery.

n_t : the number of streams to be delivered at time t

s_t : the streams to be delivered at time t

a_t : the number of streams allowed for available bandwidth

Here, (s_t, n_t) can be determined after performing SUBSCRIBE procedure of SRSM-L for all recovery requested receivers at time t.

The algorithm for the SRSM-B method is shown in Algorithm 2. Depending on the time-variant number of streams (a_t) allowed by the sender and the required number of streams (n_t), the proposed method decides whether stream merging should be performed or not. If the bandwidth requirement for the recovery stream delivery fits into the limit of the sender's available bandwidth, no further stream merging is necessary. This situation is described in line number (3) and (4). Here, we assume that the sender's allowed number of streams is always greater than 1.

For the situation where the bandwidth requirement is beyond the sender's limit, more than one stream from the (s_t) are merged into the new stream. Here, the new stream is denoted as the merged stream (*mergestream*). Stream merging is performed among the streams except the original stream ($stream_{t(ori)}$). For merging other streams except the original stream, both the required number of streams and allowed number of streams are reduced by one because we assume that the original stream is a merged stream. Then, the number of streams to be merged into a merged stream (m) is decided by dividing the reduced number of streams (q) by the reduced allowed number of stream ($a_t - 1$). The conditional statements are described from line number (7) to line number (10).

Other streams to be merged are chosen according to the amount of the data already received. In general, a stream which has minimum amount of data received is merged so that the latency of receivers is not much larger than their acceptable latencies. We refer the stream with minimum amount of data already received at time t as $stream_{t(min)}$. To achieve the safe delivery, the skip rate and bandwidth of the merged stream is set using the minimum skip rate and minimum bandwidth among all the streams to be merged, respectively. Then, the scheme modifies the latency of receivers from the streams to be merged into a merged stream. The

Table 2.3: Common Simulation Setup

Parameter	Value
Simulation time	1000 (unit time)
Number of receivers	100
Acceptable skip rate of the receiver-p (r_p)	2.0
Acceptable bandwidth of the receiver-p (b_p)	1.0
Acceptable latency of the receiver-p (l_p)	200 (unit time)
Number of streams allowed by the sender (a_t)	2
Number of trials	1000

modification of the latency is described in line number (22). The line numbers from (11) to (29) shows conditional statements for merging multiple streams. In Algorithm 2, the initialization of a merged stream is described from line number (26) to (29). Finally, the original stream is added as a merged stream at line number (30).

2.5 Evaluation by Simulations

We conducted the simulation evaluations for our proposed methods. We evaluated the number of streams, the average number of streams, the number of receivers that exceed acceptable latency and average of maximum latency for two different failure/recovery patterns.

We evaluated the methods for two failure/recovery simulation scenarios: 1) Poisson scenario and 2) Gaussian scenario. Poisson scenario corresponds to a random failure situation and Gaussian scenario corresponds to a burst failure situation. We selected these scenarios because they are typical failure situations. For example, the errors of smartphones (e.g. starting/stopping an application) in a wide area may follow the Poisson scenario and the errors when the receivers in a train that goes into tunnel sometime may follow the Gaussian scenario.

2.5.1 Simulation Setup

The common simulation setup is shown in Table 2.3, where p is a receiver in the simulation.

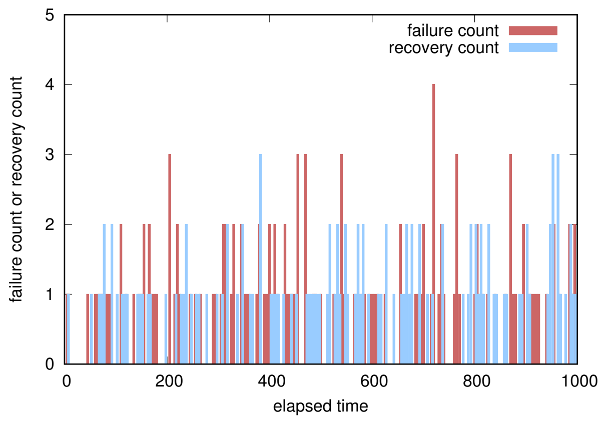


Figure 2.6: Loss and Recovery Model of the Second Trial (Poisson)

We used a notion of virtual ‘unit time’, which can correspond to one data block transmission time in the simulations. The acceptable bandwidth ($b_p = 2$) is selected because we assume IoT applications, in which plenty of network bandwidth is not available. The simulation time (= 1000 unit time) is decided as an enough duration for the stable simulation. The number of receivers (= 100) is an enough number that we can reproduce the congested situation. $r_p = 2$ and $l_p = 200$ are selected as typical parameters to see the basic behavior. Same as the unit time, both r_p and l_p can be translated for various situations. If we want to translate 50 milliseconds as 1 unit time, then the acceptable latency l_p (= 200 unit time) is equal to 10 seconds. To see the basic performance, we assume all receivers have the same requirements for the recovery.

In the Poisson scenario, a receiver encounters failure multiple times within the simulation time because the number of receivers is limited in the simulation. The average number of failure occurred within a unit time is set as 8. Fig. 2.6 shows the failure/recovery pattern of the second trial for the Poisson scenario. The x-axis represents the elapsed time and y-axis represents the number of failure or recovery.

In the Gaussian scenario, a receiver can encounter a failure at most one time within the simulation time. In all trials, we assume half of the receivers encounter failure around 500 unit time following the Gaussian distribution. The standard deviation of the Gaussian distribution is set as 50. The failure interval also follows

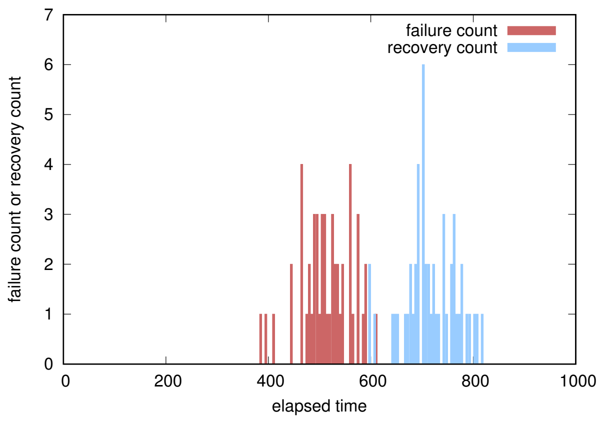


Figure 2.7: Loss and Recovery Model of the Second Trial (Gaussian)

the Gaussian distribution. The average failure interval (fi) is set from fi=100 to 200 by the increment of 10. The standard deviation of the average failure interval is set as 10. Fig. 2.7 describes the assumed loss and recovery model of the second trial for the Gaussian scenario.

2.5.2 Comparison methods

The same notations shown in Table 2.1 and Table 2.2 are used to describe the parameters of the comparison methods. In the Piggybacking, a slow stream is generated when no slow stream exists within a time window W measured in frames. Otherwise, a fast stream is generated. The slow stream means the stream that are delivered at a lower rate while the fast stream is the one delivered at a faster rate. The idea is that the fast stream will eventually catch up with the slow stream. As the slow streams are generated for each time window W, the senders have to use more bandwidth. However, the receivers do not need to have large buffer storage. Therefore, the simple Piggybacking scheme can be used for IoT applications. And we consider the Piggybacking as a method to be compared with the proposed methods. For simulation, we assume that the slow stream has R_s =1.0 and B_s =1.0 while the fast stream delivers with R_s =2.0 and B_s =1.0. The window size W is set as 40. Another comparison method is Coincidental merging in which the receiver joins the existing stream when it has exactly same parameter values of the existing stream.

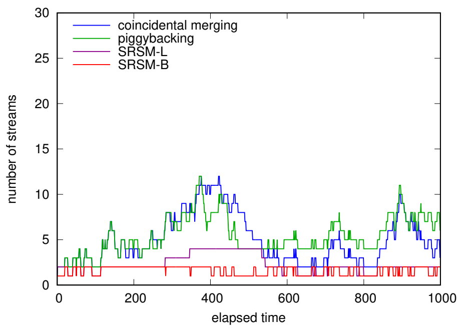


Figure 2.8: Number of streams in the Second Trial (Poisson)

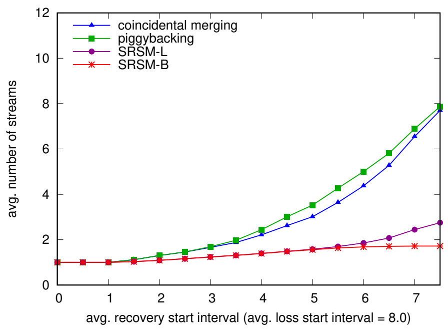


Figure 2.9: Average number of streams (Poisson)

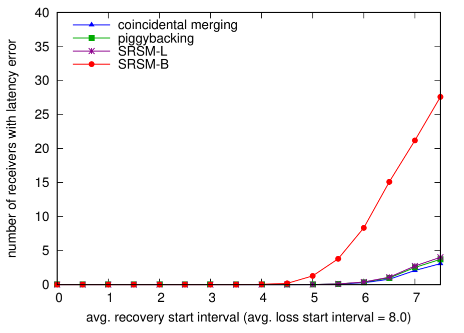


Figure 2.10: Number of receivers that exceed acceptable latency (Poisson)

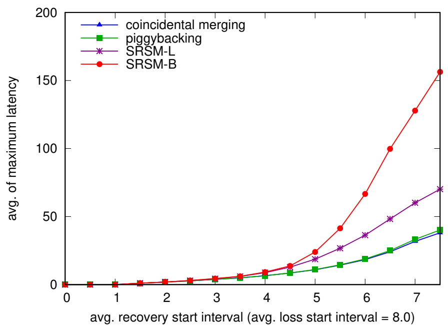


Figure 2.11: Average of the maximum latency (Poisson)

2.5.3 Simulation Results in the Poisson Scenario

Figs. 2.8-2.11 shows the simulation results for the Poisson scenario. We denote λ as the average number of failures occurs within a unit time and μ as the average number of recoveries within a unit time. For the Poisson scenario, we set ($\lambda = \frac{1}{8}$) and ($\mu = \frac{1}{h}$) to define the average number of failure and recovery respectively. The value of (h) is set from 0.5 to 7.5 with the incrementing value 0.5. The comparison of the number of streams resulted from the second trial is shown in Fig. 2.8. The parameters for the second trial are ($\lambda = \frac{1}{8}$) and ($\mu = \frac{1}{7.5}$). Fig. 2.9 shows the average number of streams by changing the average interval between recoveries, which corresponds to $\frac{1}{\mu}$. Fig. 2.10 shows the number of receivers that exceeds the acceptable latency and Fig. 2.11 shows the average of the maximum latency for all simulation trial.

According to these results in Poisson scenario, the SRSM-L method showed good performance than the Piggybacking and Coincidental merging, when the average interval between recoveries is large, which means the number of receivers in simultaneous failure is large, in terms of the number of streams. The performance of the Piggybacking is worse around the average interval value between 4 and 8. That is because more stream generation occurs in the Piggybacking and the random failure gives disadvantage for it. The average number of streams (i.e. network bandwidth used by the sender) in the SRSM-L method is reduced about 52% compared with the Piggybacking and Coincidental merging when the average interval between recoveries is 7.5. On the other hand, as expected, the number of streams of the SRSM-B method did not exceed 2. However, as shown in Fig. 2.10, the number of receivers which exceeded the acceptable latency increases. As the Fig. 2.11 shows, the average latency increases to around 250 when the loss start interval = 7.5.

As a summary, in the Poisson scenario, we can say the SRSM-L method can keep good performance even when there are frequent failures and recoveries. The SRSM-L method showed balanced results which keep the bandwidth usage and the delivery latency small. About the SRSM-B method, we confirmed that the method could keep the number of streams constant, but increased delivery latency. Therefore, the SRSM-B method is suitable only when the application does not need to limit the latency in the Poisson scenario.

2.5.4 Simulation Results in the Gaussian Scenario

Figs. 2.12-2.15 shows the simulation results for the Gaussian scenario. For the Gaussian scenario, the simulation is performed using different average failure interval (fi) where (fi = 100 to 200) (unit time) with standard deviation 10.

Fig. 2.12 shows the result of stream count for the second trial. The parameters for the second trial are mean failure start time (500) with standard deviation (10), and average failure interval (200) with standard deviation (10). Depending on the average length of the failure interval, the recovery start time for the receivers is slightly different. As a result, more recovery stream generation occurs nearly at the same time. The comparison of all methods in terms of the average number of

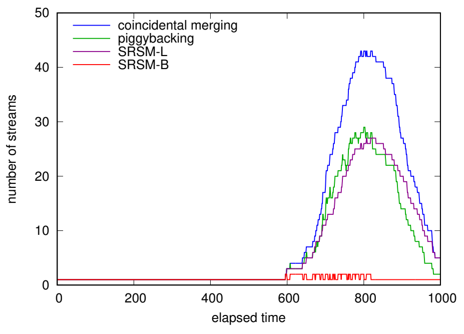


Figure 2.12: Number of streams in Second Trial (Gaussian)

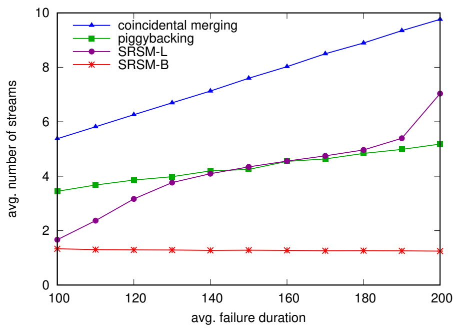


Figure 2.13: Average number of streams (Gaussian)

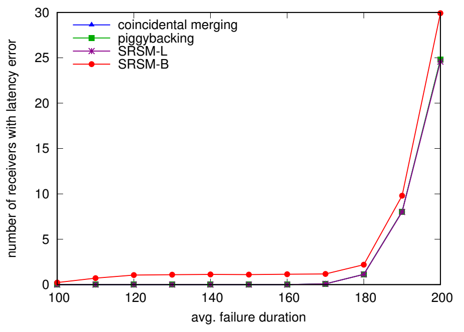


Figure 2.14: Number of receivers that exceed acceptable latency (Gaussian)

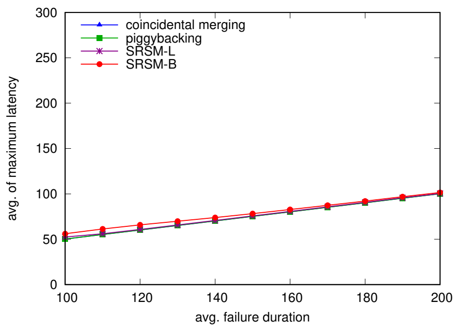


Figure 2.15: Average of the maximum latency (Gaussian)

streams is illustrated in Fig. 2.13. Fig. 2.14 shows the number of receivers that exceeds the acceptable latency and Fig. 2.15 shows the average of the maximum latency for all simulation trials.

According to the experimental results described in Fig. 2.12, the Coincidental merging showed worst performance in terms of the number of streams. The Piggybacking showed better performance than Coincidental merging. It showed similar performance with the SRSM-L method because simultaneous errors can be accommodated to the merging stream in the Gaussian scenario. the number of streams increases in the SRSM-L method as the average length of the failure interval increases. That is because the number of long failure receivers, in which the failure interval exceeds 200, increases as shown in Fig. 2.14. The average number of streams in the SRSM-L method is nearly equal to that of the Piggybacking when the difference is from 40 to 80.

As expected, the SRSM-B method could keep the number of streams no more than 2 during the simulation time. The average number of receivers that exceed the acceptable latency and the average max latency becomes larger when the fi becomes larger in the all methods. As shown in the Fig. 2.14 and Fig. 2.15, the SRSM-B method showed larger latency than other methods. However, the increase in the latency was small.

As a summary, in the Gaussian scenario, we can say the SRSM-B method could keep good performance. The SRSM-B method could keep the max bandwidth usage constant and the delivery latency overhead small. SRSM-L, could not keep the number of streams small, when the difference is large, that means the failure duration is longer than the threshold l_p.

2.6 Conclusion

Aiming at the bandwidth reduction, we proposed efficient lost sensor data delivery methods for sensor data multicasting called “Latency-aware synchronized recovery stream merging (SRSM-L)” and “Bandwidth-dependent synchronized recovery stream merging (SRSM-B)”, which are suitable for IoT applications. We evaluated the performance in two simulation scenarios that correspond to the random failures and the burst failures. Simulation results are shown in terms of the number of streams, the average number of streams, the number of receivers that exceeds the acceptable latency and the average of the maximum latency.

According to the evaluation results, we found that the performance of the SRSM-L method can save about 52% network bandwidth on the sender in the frequent random failure situation, comparing with existing schemes, keeping the acceptable latency. We also found that in the burst failure situation, the SRSM-B method shows the best performance even when the failure duration is long. It could keep the maximum number of streams constant and the latency overheads small. In the future work, we will do more detailed evaluations to confirm whether our scheme is applicable for more applications or situations. Then we will implement the proposal method on an actual pub/sub multicasting platform for reliable IoT applications. In addition, further improvement of delivery load reduction mechanism, the recovery stream generation scheme using distributed caches on the network, dynamically applying the appropriate method (SRSM-L or SRSM-B) according to the observations and feedback, also form part of our future work.

Chapter 3

Certificate-less Multi-receiver Encryption with Authentication based on ECC

3.1 Introduction

In some IoT applications related to healthcare, smart homes and group communications, sensitive data is exchanged among multiple users. For example, pedestrians who feel concerns of their health around a station send vital data, which are generated by body sensors attached to smartphones, to nearby doctors or nurses for the purpose of improving experience, getting advice or healthcare service. In such applications, for quick response to the patient, the edge computing is attractive architecture. In the edge computing, the sensor data are sent to the smartphone, and data processing is done on the edge computer or the smartphone of the doctor. Therefore, the smartphones need to support lightweight algorithms in order to interoperate with the resource-constrainted sensors. And a lightweight multi-receiver encryption scheme to ensure confidentiality, integrity, and authenticity is necessary to provide not only security but also efficiency.

To securely exchange information with fast computing speed, symmetric encryption scheme can be used. However, it introduces security requirements such as authentication of parties in key agreement protocol or secure and integrityassured key distribution to prevent man-in-the-middle attacks and other attacks. Without the use of public-key cryptography, symmetric key scheme is not sufficient to get secure communication features such as confidentiality, authentication, integrity, and non-repudiation. Although the conventional public key infrastructure (PKI) is widely used in the current ICT systems, it is not suitable for resource-constrained devices due to its certificate management problem [63]. To avoid the certificate management problem of public key infrastructure, in identity-based cryptography introduced by Shamir [64], unique strings such as identities are used as public keys. However, a party called private key generator (PKG) exists to generate system parameters and private keys for all users. As a result, private key generator knows all users' private keys. This problem is called key escrow problem. Key escrow problem introduces some serious disadvantages such as communication information exposure.

To simplify the certificate management of traditional public key cryptography and key escrow problem of identity-based public key encryption, the concept of certificate-less public key cryptography has been proposed by Al-Riyami and Paterson [65]. In certificateless public key cryptography (CL-PKE), key generation process is split between PKG and the users. Key Generation Center (KGC) issues the partial private key and the user generates own public key and private key pair. In encryption, the sender uses the public system parameters of KGC, the public keys of the receivers, and the plaintext message. To decrypt the ciphertext, the receivers require both partial private key and private key. If we assume that KGC is honest but curious, KGC cannot decrypt exchanged messages by knowing only the partial private key. In this chapter, we define *semi-trusted third party* to represent an honest but curious KGC. In other words, KGC knows the master secret key to generate partial private keys and it may misbehave. However, it is not allowed to replace any public keys. In this way, key escrow problem is avoided. Moreover, certificateless public key encryption eliminates the use of certificate as public key of user is generated using the parameters given by KGC. The overview of certificate-less public key encryption is shown in Fig. 3.1.

In devices with sensitive personal data, no trust exists on others. Full control is in the hand of device owner. In sharing sensitive data among own controlled devices, it is necessary to ensure trustworthiness of users. Although third party

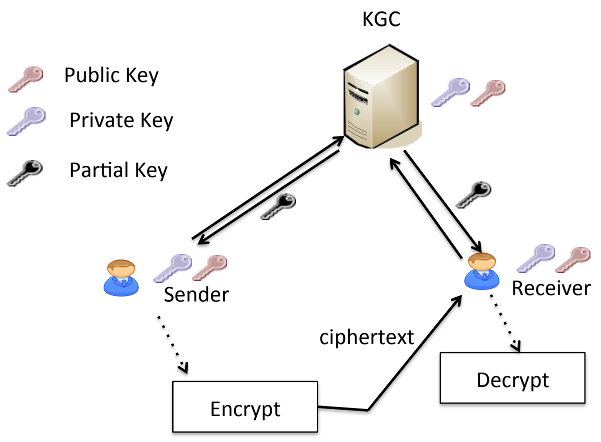


Figure 3.1: Overview of certificateless public key encryption

involvement may exist to prove valid users, the key escrow problem should be avoided so that users can control data access directly. To be able to share the sensitive data among own controlled devices, certificate-less public key encryption is an optimal solution. Therefore, we adopt certificate-less public key encryption into our multi-receiver environment. However, for this, we should reduce the computational load for both the sender and the receiver and provide the necessary security properties.

In this chapter, we propose a certificate-less multi-receiver encryption scheme using elliptic curve cryptography. To ensure authentication, the proposed scheme provides implicit user authentication and source authentication. Moreover, the proposed scheme offers message integrity and replay attack prevention as the necessary security properties for secure data exchange. In addition, pairing operations that are computationally expensive are eliminated to achieve lower latency of encryption and decryption. The latency of encryption is the time between the initial request and the reply of the corresponding ciphertext. In a similar fashion, the latency of decryption is the time for decrypting the ciphertext. In this chapter, we provide security proof for the proposed scheme based on the intractability of the Elliptic Curve Discrete Logarithm problem (ECDLP). According to the computational cost comparison and experimental results, we confirm that the proposed scheme achieves multi-receiver encryption scheme with better efficiency and more security properties compared with existing schemes.

The remainder of this chapter is organized as follows. In Section 3.2, we introduce the related work. Preliminaries are introduced in Section 3.3. We present our proposed scheme in Section 3.4. In Section 3.5, we show the results of the experiments, and in Section 3.6, we give the security proofs of our proposed scheme. Finally, in Section 3.7, we summarize this chapter.

3.2 Related Work

For multi-receiver setting, several identity-based encryption schemes and certificateless encryption schemes have been proposed.

Although identity-based encryption schemes have key escrow problem, some identity-based encryption schemes try to avoid key escrow problem. In identity-based multi-receiver encryption schemes ([66]-[77]), key escrow problem exists. In [77], key escrow problem exists although source authentication property is provided for multi-receiver setting. In [78], no key escrow problem exists as users generate own key pair. Although key generation introduces some load on the sender, it can avoid key escrow problem. In terms of computational cost, the scheme [78] achieves better efficiency than identity-based multi-receiver encryption scheme proposed in [66] because it reduces one pairing operation not only for encryption but also for decryption. However, in [78], the sender uses only the public keys of the receivers for encryption. As a result, the adversary can easily impersonate as the authorized receiver in communication of unknown devices in large and highly dynamic environment. There are two options for public key validation: using central public key directory service or explicit key validation. In the former case, communication overhead exists. Computational overhead about two pairing operations will be required in the latter case. Moreover, the scheme does not consider source authentication and replay attack prevention.

Certificateless public key scheme has been proposed to avoid key escrow problem. However, most of the certificateless public key schemes use computation expensive bilinear pairing operations [65, 79]. For better performance, certificateless public key encryption schemes without pairing have been proposed in [80] and [81]. To achieve stronger security model than that of [80], [81] has been proposed. In [81], a pair of ciphertext is required for a single receiver. As a result, the ciphertext length will increase as the number of receivers becomes larger. Moreover, source authentication and replay attack prevention are not considered in both schemes. To achieve certificateless multi-receiver encryption with source authentication, [79] and [83] have been presented. In [79], expensive pairing operations are required for encryption and decryption. And the number of pairing operation is directly proportional to the number of receivers in multi-receiver setting. Although [83] avoids pairing operation in encryption, it requires two pairing operations for decryption and source authentication. Moreover, its ciphertext size is twice the ciphertext size of identity-based signcryption scheme [77]. To resist Type-I adversary attack in [83], an enhanced scheme is proposed in [84]. In [84], the security weakness in [83] is solved by eliminating randomness reuse in the computation of a parameter for all receivers. As a result, more pairing exponentiation are required for encryption. Moreover, decryption needs one more pairing operation and replay attack prevention is not considered. Existing certificate-less multi-receiver encryption scheme has been presented in [85]. In [85], computation expensive pairing operations are used in both encryption and decryption. In addition, the scheme does not consider source authentication and replay attack prevention. Therefore, in this chapter, we propose a multi-receiver encryption scheme that considers source authentication, replay attack prevention, and message integrity. In addition, the proposed scheme does not require certificates, and pairing operations.

3.3 Preliminaries

In this section, we describe elliptic curve cryptography, bilinear pairing, Schnorr signature scheme, security requirements for group communication, and the framework of the proposed scheme,.

3.3.1 Elliptic Curve Cryptography

Elliptic curve cryptography (ECC) is an approach to public-key cryptography based on the elliptic curves over finite fields. ECC was proposed by Victor S.Miller

Table 3.1: Key Size (bits) Comparison for equivalent security level

Symmetric	RSA and Diffie Hellman	Elliptic Curve
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

[86] in 1986 and by Neal Koblitz in 1987. An elliptic curve E over F_p is defined by an equation of the form $y^2 = x^3 + ax + b$ where $a, b \in F_p$, p is a large prime, and F_p is a finite field containing p elements. Elliptic curves are widely used in cryptography due to its smaller key size and faster scalar multiplication. For example, a 160-bit elliptic curve key provides the same security level while RSA uses 1024-bit [89]. By using ECC, the system can save the power, bandwidth and storage. The comparison of key size is shown in Table 3.1.

i) Scalar Multiplication
Let P be the point on E. Then, the scalar multiplication sP means adding P for s-times.

ii) Elliptic Curve Discrete Logarithm Problem
Let P and $Q = xP$ be two points on elliptic curve E. Given P and Q, the elliptic curve discrete logarithm problem (ECDLP) is to find x. Elliptic curve parameters such as underlying field or group determine the difficulty of the problem. ECDLP is assumed to be intractable for certain elliptic curve parameters. The security of elliptic curve cryptography is based on the intractable level of the elliptic curve discrete logarithm problem (ECDLP).

3.3.2 Bilinear Pairing

A pairing is a map $e : G1 \times G2 \rightarrow GT$ where $G1, G2$ are cyclic group of the same prime order. $G1$ and $G2$ are additive groups and GT is a multiplicative group. In symmetric pairing, $G1$ and $G2$ are equal. When $G1 \neq G2$, the pairing is said to be asymmetric. Let $P \in G1, Q \in G2$ be generators of $G1$ and $G2$ respectively. A pairing satisfies the following conditions.

i) Bilinearity: $\forall a, b \in Z : e(aP, bQ) = e(P, Q)^{ab}$

ii) Non-degeneracy: $e(P, Q) \neq 1$

iii) Computability: e has to be computable in an efficient manner

Pairing-based cryptography uses the bilinear pairings [88]. And the pairing operation is the time and power inefficient one for resource-constrained devices. For example, in the testbed 3 using HTC A9191 with android version 2.2, average time for scalar multiplication is 265.6 (ms) while the bilinear pairing requires 491.2 (ms) for type-a pairing in super-singular elliptic curve group [94]. If the number of pairing operations becomes fewer, the scheme becomes more efficient. The security of the pairing-based cryptography depends on the intractable level of the elliptic curve discrete logarithm problem.

3.3.3 Schnorr Signature Scheme

For efficient identification and signature, Claus-Peter Schnorr proposed the Schnorr signature scheme in [90]. The Schnorr signature scheme uses a group G in which the discrete logarithm problem is hard. Let G be a group of prime order q, P be generator, and $H : \{0, 1\}^* \in Z_q$ be one-way hash function. It consists of the following algorithms.

- Key generation: Generate public key $pk = (P, Q = sP)$ by choosing private key $s \in Z_q$.
- Signing: For signing message $M \in \{0, 1\}^*$, choose $r \leftarrow Z_q$ randomly. Then, compute $R = rP$, $e = H(M||R)$ and $t = r + se \, mod \, q$. The signature is (R, t).
- Verifying: Compute $e(M||R)$. If $tP = R + eQ$, the signature is verified.

The security of Schnorr signature scheme is based on intractable one-way hash function and discrete logarithm problem.

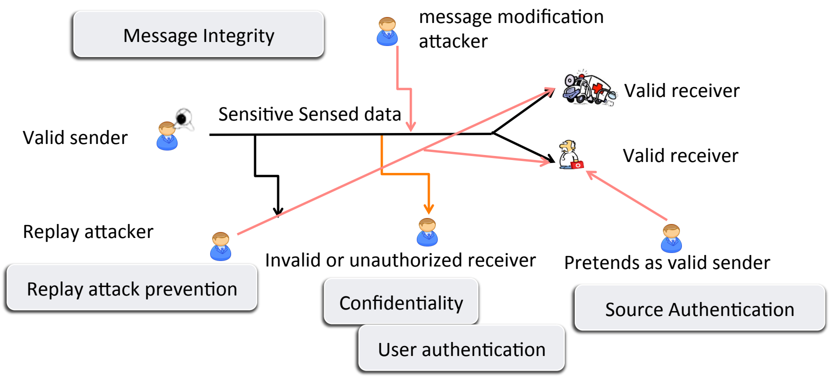


Figure 3.2: Security Requirements in Group

3.3.4 Security Requirements for Group Communication

For the group communication, there are basic security requirements. For example, as shown in Fig. 3.2, there are attackers who try to modify the message, replay the valid message, reveal the sensitive data, and pretend as a valid sender. Therefore, confidentiality, message authentication, and replay attack prevention are required.

- Confidentiality: Confidentiality is the process that prevents sensitive information from unauthorized receivers. Only authorized receivers can reveal the information.

- Message Authentication: Source authentication is the process that the receivers can verify the source of the message. Message integrity is the assurance that the message has not been altered in transit. Message authentication is a property that can prove both source authentication and message integrity.

- Replay Attack Prevention: A replay attack is a kind of attack in that the attacker eavesdrops valid message and sends that message repeatedly to make requester in busy mode. It can be prevented by adding session identifiers to the broadcasted message.

3.3.5 Framework of the proposed scheme

A proposed scheme consists of five polynomial time algorithms.

- Setup: Semi-trusted third party runs this algorithm to generate public system parameters *params* and its master secret key s.
- Set-Key-Pair: For key escrow problem avoidance, all users (senders and receivers) run this algorithm to generate public key and private key.
- Partial-Private-Key-Extract: Semi-trusted third party generates identity-based partial private keys for all users based on Schnorr signature scheme.
- Encrypt: Sender runs this algorithm to encrypt the message M by using *params*, public key of semi-trusted third party and target receivers' public keys. To maintain message authentication and replay attack prevention properties, the sender generates signature using its own private key and partial private key. Finally, ciphertext C is given as output.
- Decrypt: Receiver runs this algorithm that takes C, *params*, partial private key and private key as input to get back plaintext message M.

3.3.6 Indistinguishability under adaptive chosen ciphertext attack (IND-CCA2)

To prove the strength of the cryptosystem, we have to assume some class of capability of the attacker (the adversary) and how does the attacker break the system. For the formal definitions of security of the cryptosystem, Indistinguishability has been introduced and a game between the adversary and the cryptosystem is defined [100]. Indistinguishability of encryptions means that an adversary cannot distinguish pairs of ciphertexts given an encryption of a message randomly chosen from a two-element message space determined by the adversary. Instead of randomly guessing message choice, a game allows the adversary to learn some information except challenged ciphertext. The adversary $\mathcal{A}$ is considered to have an advantage if $\mathcal{A}$ can distinguish the chosen ciphertext with a probability significantly greater than that of random guessing $\frac{1}{2}$. The scheme is considered secure if no adversary has a non-negligible advantage in winning the game.

We assume that the proposed scheme uses elliptic curve parameters that are strong enough for discrete logarithm problem. For the class of the capabilities of the adversary, Ciphertext Plaintext Attack (CPA), Chosen Ciphertext Attack (non-adaptive) (CCA1), and Chosen Ciphertext Attack (adaptive) (CCA2) are commonly considered. In the game for CPA attack, the adversary is not allowed to query Decrypt oracle. The CCA1 adversary can query Decrypt oracle only up until it receives the challenge ciphertext. Among these, CCA2 is considered the stronged one. Therefore, we will consider only CCA2 for further discussion. A proposed scheme is "Indistinguishability of encryptions under the adaptive chosen ciphertext attack" (IND-CCA2) secure if no polynomial-time adversary $\mathcal{A}$ has a non-negligible advantage in the following game played against the challenger C under hard ECDLP assumption.

- Setup: The challenger C runs the Setup algorithm to generate public system parameters *params* and master secret key s. It gives public system parameters *params* to adversary $\mathcal{A}$. Master secret key is kept secret if adversary $\mathcal{A}$ is Type-I adversary. In the case of Type-II adversary $\mathcal{A}$, master secret key is given to $\mathcal{A}$.

- Phase 1: Challenger C chooses target identities $ID_i^* = \{ID_1^*, ID_2^*, ..., ID_n^*\}$, target sender ID_u and sends them to $\mathcal{A}$.

- Phase 2: $\mathcal{A}$ can access the following queries for $q_1, q_2, .., q_{de}$ where q_{de} is one of
  - Hash queries: The challenger C returns the results of hashed operations for inputs given by $\mathcal{A}$.
  - Oracles: The adversary $\mathcal{A}$ can issue requests to corresponding oracles with the restricted adversary behavior.

- Challenge: Adversary $\mathcal{A}$ chooses two plaintext messages (M_0, M_1) and sends them to challenger C, with restriction that M_0, M_1 are two distinct messages of the same length. $\mathcal{A}$ is allowed for all the queries in Phase 2. C then randomly selects $\alpha \in \{0, 1\}$ and ciphertext C^* is generated using ID_i^*, ID_u and M_α.

- Phase 3: $\mathcal{A}$ can make the same queries in Phase 2, except Decrypt oracle with $C^* = C$ and $ID_i \in ID_i^*$.

- Guess: Finally, $\mathcal{A}$ outputs $\alpha' \in \{0, 1\}$ and wins the game if $\alpha' = \alpha$. The adversary $\mathcal{A}$ is defined as $IND - CCA2$ adversary.

The advantage of $\mathcal{A}$ to win the game is

$$Adv^{IND-CCA2}(\mathcal{A}) = |Pr[\alpha' = \alpha] - \frac{1}{2}|$$

Adversary $\mathcal{A}$ breaks IND-CCA2 with (ti, q_{de}, ϵ) if adversary $\mathcal{A}$'s advantage ϵ is non-negligible after q_{de} queries within running time ti. If no adversary breaks the scheme with (ti, q_{de}, ϵ), then the scheme is (ti, q_{de}, ϵ)-IND-CCA2 secure.

In the above game, the adversary is allowed to perform a polynomially bounded number of encryptions, decryptions and other operations. Then, the adversary submits two distinct chosen plaintexts (M_0, M_1) to the challenger. The challenger selects a bit $\alpha \in \{0, 1\}$ uniformly at random and sends the ciphertext $C = E_K(M_\alpha)$ to the adversary where $E_K(M_\alpha)$ is the encryption of message M_α under the key K. The adversary may perform any additional encryptions or other operations. The adversary is allowed to perform Decrypt oracle based on the previous chosen ciphertext queries even after it has received a challenge ciphertext. Finally, the adversary must output a guess for the value of α within a polynomial number of time steps. The scheme is IND-CCA2 secure if no adversary has a non-negligible advantage in winning the game.

3.3.7 Security Model

In certificateless cryptography, it is necessary to model the attackers or the adversaries who can replace public keys and/or who know master secret because attacks may come from outside adversaries and semi-trusted third party defined in subsection 3.1. In this section, we describe two types of adversaries that need to be considered in any certificateless encryption scheme, the oracles that they can access, and the restricted behavior for them.

3.3.7.1 Common Oracles

To ensure the security of the proposed scheme, a game is played between an adversary $\mathcal{A}$ and a challenger C. In the game, the adversary can access the following oracles.

- CreateUser: The oracle accepts identity as input and returns the corresponding public key. If the public key for the identity does not exist in the user list, it runs Set-Key-Pair and Partial-Private-key-Extract algorithms to generate public key, private key and partial private key. And then, it returns the public key.
- Request-Public-Key : The oracle accepts identity as input. It returns the corresponding public key.
- Extract-Private-Key: This oracle accepts an identity as input and outputs the corresponding private key. The restriction of the oracle is that the adversary cannot request Extract-Private-Key if he or she has already replaced the public key for identity.
- Extract-Partial-Private-Key: This oracle takes an identity as input and returns the corresponding partial private key. The restriction of the oracle is that the adversary cannot request Extract-Partial-Private-Key if he or she has already replaced the public key for identity.
- Decrypt: The oracle accepts an identity and a ciphertext as input. Using private key and partial private key corresponding to the given identity, the oracle decrypts the ciphertext and outputs the corresponding plaintext.

3.3.7.2 Type-I Adversary

Type-I adversary models the attacks by anyone except the legitimate receivers or the KGC. Therefore, the adversary does not possess the master secret key. The adversary can access all of the common oracles described in subsection 3.3.7.1. Moreover, the adversary is allowed to replace public key. Therefore, Type-I adversary can also access the following oracle.

- Replace-Public-Key: This oracle allows the adversary to replace the public key of identity with any valid values of its own choice.

The following restrictions exist for Type-I adversary.

- Adversary cannot access Extract-Partial-Private-Key oracle for the target identities and target sender at any point.
- Adversary cannot access Extract-Private-Key oracle for the target identities and target sender at any point.

3.3.7.3 Type-II Adversary

Type-II adversary models the attacks to break the confidentiality of the scheme. An honest but curious KGC who knows master secret key represents Type-II adversary. The adversary is not allowed to replace any public key. Type-II adversary can access the common oracles described in subsection 3.3.7.1.

The following restrictions exist for Type-II adversary.

- Adversary cannot access Extract-Private-Key oracle for the target identities and target sender at any point.

3.4 Proposed Scheme

In this section, we will present the proposed scheme with security notations in detail. The security notations and their descriptions are shown in Table 3.2.

3.4.1 Overview of CLAME

Our proposed scheme is denoted as CLAME (Certificate-less Multi-receiver Encryption with Authentication). In CLAME, semi-trusted third party has to generate the public system parameters to be shared by all users. In addition, semi-trusted third party acts as an authority who issues the proof for the validity of the users by generating the partial private key. All users need to keep the partial private key secret. Besides, all users have to generate own public key and private key. When the data or the message is encrypted, the sender uses the public system parameters, public keys of the receivers, own partial private key and private key. Then, the receivers try to decrypt the ciphertext. In the decryption, the receiver needs to use own private key and partial private key. CLAME avoids pairing operations in both encryption and decryption.

Table 3.2: Notations and Descriptions

Notation	Description
p, q	Large primes
E	Elliptic curve with domain parameters over F_p
P	Point of order q on E
G_1	Cyclic group generated by P
Pub_s	Public Key of semi-trusted third party
s	Master Private Key of semi-trusted third party
$params$	Public system parameters
pk_i	Public key of i-th user
sk_i	Private key of i-th user
X_i, d_i	Partial private key for i-th user
$_u$	Symbol to represent the parameters of sender
H_1, H_2, H_3, H_4	Cryptographic one-way, collision-resistant hash functions
E_K	Symmetric encryption using key K
D_K	Symmetric decryption using key K
$M \in \{0,1\}^*$	Message
$t \in \{0,1\}^*$	Current Time
k, k_0	Bit lengths

3.4.2 Algorithms

Proposed scheme consists of the following algorithms.

- Setup: Semi-trusted third party runs this algorithm to generate public system parameters *params* and master secret key ($s \in Z_q$). Master secret key is kept secret.
  The public parameter *params* consists of $\{E, k0, k, P, H_1, H_2, H_3, H_4, Pub_s = sP\}$ where $H_1 : \{0,1\}^* \times G_1 \times G_1 \rightarrow Z_q, H_2 : G_1 \rightarrow \{0,1\}^{k0}, H_3 : \{0,1\}^* \rightarrow Z_q, H_4 : \{0,1\}^{k0} \rightarrow \{0,1\}^k$.

- Set-Key-Pair: Each user (i) runs this algorithm to generate key pair. Firstly, user selects $sk_i \in Z_q$ randomly. To obtain partial private key, $(ID_i, P_i = sk_iP, P_{i1} = sk_iPub_s)$ is sent to semi-trusted third party. After getting partial private key from semi-trusted third party, user performs the following steps.
  - Computes $P_i^{'} = P_i + X_i$ and $P_i^{''} = h_i^{-1}P_i^{'}$ where $h_i = H_1(ID_i, P_i, P_i^{'})$.

- Declares $pk_i = (ID_i, P_i, P_i^{'}, P_i^{''})$.

- Partial-Private-Key-Extract: Semi-trusted third party generates partial private key by performing the following steps.

  1. Semi-trusted third party checks the validity of user's public key by checking whether P_{i1} is equal to sP_i or not. If not, semi-trusted third party rejects for partial private key generation.
  2. Additional verification process is done using user identity and other additional verification proofs.
  3. After the user has been successfully verified, semi-trusted third party performs the following steps.
     - Selects $x_i \in Z_q$ uniformly at random.
     - Computes $X_i = x_i P$, $CP_i = P_i + X_i$, and $d_i = H_1(ID_i, P_i, CP_i)s + x_i \, mod \, q$.

  Then, (X_i, d_i) is given to user via secure channel.

- Encrypt: To encrypt message M for n-receivers ($j = 1, 2, ...n$), the sender chooses $r \in Z_q$ and $\sigma \in \{0, 1\}^k$ randomly. Then, it computes

$$\begin{aligned}
Y &= rP \\
m &= H_3(M||\sigma||L||t||Y) \\
a &= h_u^{-1} d_u + msk_u + r \, mod \, q \\
Z &= mP \\
h_j &= H_1(ID_j, P_j, P_j^{'}) \\
U_j &= mh_j(Pub_s + P_j^{''}) \\
V &= \sigma \oplus H_2(Z) \\
K &= H_4(\sigma) \\
W &= E_K(M)
\end{aligned}$$

  Then, ciphertext $C = (a, U_1, U_2, ..., U_n, V, W, Y, L, t)$ is sent to the receivers where L is a label describing the association of each receiver and U_j.

- Decrypt: Each receiver performs the following steps to decrypt the ciphertext C.

  1. Find corresponding U_j from label L.
  2. Compute $Z' = (d_j + sk_j)^{-1}U_j$.
  3. Compute $\sigma' = V \oplus H_2(Z')$.
  4. Compute decryption key $K' = H_4(\sigma')$.
  5. Decrypt the ciphertext to get back the plaintext message $M' = D_{K'}(W)$.
  6. Compute $m' = H_3(M' \| \sigma' \| L \| t \| Y)$ and check whether Y is equal to $aP - ((m' - h_u^{-1})P_u + P_u'' + Pub_s)$ or not. If not, return "reject".
  7. Check whether U_j and $m'h_j(Pub_s + P_j'')$ are equal or not. If they are not equal, reject the ciphertext. Otherwise, return M' as the plaintext.

3.4.3 Security Correctness

To get back the plaintext message, the authorized receivers can decrypt the ciphertext by using own partial private key and private key. For example, the receiver-j can check the correctness of the proposed scheme as follows:

$$\begin{aligned}
Z' &= (d_j + sk_j)^{-1}U_j \\
&= \frac{1}{(d_j + sk_j)}U_j \\
&= \frac{mh_j(Pub_s + P_j'')}{(H_1(ID_j, P_j, CP_j)s + x_j) + sk_j} \\
&= \frac{mH_1(ID_j, P_j, P_j')(Pub_s + P_j'')}{(H_1(ID_j, P_j, CP_j)s + x_j) + sk_j} \\
&= \frac{mH_1(ID_j, P_j, P_j')(sP + h_j^{-1}(sk_j + x_j)P)}{H_1(ID_j, P_j, P_j')s + x_j + sk_j} \\
&= \frac{m(H_1(ID_j, P_j, P_j')sP + (sk_j + x_j)P)}{H_1(ID_j, P_j, P_j')s + x_j + sk_j} \\
&= \frac{m((H_1(ID_j, P_j, P_j')s + sk_j + x_j)P)}{H_1(ID_j, P_j, P_j')s + x_j + sk_j}
\end{aligned}$$

$$\begin{aligned} &= mP \\ \sigma^{'} &= V \oplus H_2(Z^{'}) \\ &= \sigma \oplus H_2(Z) \oplus H_2(Z^{'}) \\ &= \sigma \oplus H_2(mP) \oplus H_2(mP) \\ &= \sigma \\ K^{'} &= H_4(\sigma^{'}) \\ &= H_4(\sigma) = K \end{aligned}$$

If the recovered key $K^{'}$ and the encryption key K are the same, the receivers can reveal the plaintext message. After recovering the key $K^{'}$, the authorized receivers can decrypt the ciphertext by performing the symmetric decryption using key $K^{'}$, $D_{K^{'}}(W)$.

The receivers can also check whether the ciphertext maintains the message authentication and replay attack prevention properties. To check the correctness of the message authentication and replay attack prevention, the receivers have to compute aP and $(m^{'} - h_u^{-1})P_u + P_u^{''} + Pub_s$. Then, the receivers compare the subtracted value of these values with the Y value. If they are not equal, the receivers can easily know that the attacker has modified the message or replayed the message.

$$\begin{aligned} h_u &= H_1(ID_u, P_u, P_u^{'}) \\ aP &= h_u^{-1} d_u P + msk_u P + rP \\ &= h_u^{-1}(h_u sP + x_u P) + mP_u + rP \\ &= sP + h_u^{-1} x_u P + mP_u + rP \\ &= Pub_s + h_u^{-1} x_u P + mP_u + rP \\ (m^{'} - h_u^{-1})&P_u + P_u^{''} + Pub_s \\ &= (m^{'} - h_u^{-1})P_u + h_u^{-1}(P_u + x_u P) + Pub_s \\ &= m^{'} P_u + h_u^{-1} x_u P + Pub_s \\ aP - [(m^{'} &- h_u^{-1})P_u + P_u^{''} + Pub_s] \\ &= rP = Y \end{aligned}$$

Also, each user (i) can check the validity of semi-trusted third party generated partial private key by checking whether d_iP is equal to $h_iPub_s + X_i$ or not.

$$\begin{aligned} & h_iPub_s + X_i \\ & = h_i(sP) + x_iP \\ & = (h_is + x_i) \\ & = d_iP \end{aligned}$$

If they are not equal, then the partial private key is not valid. The partial private key is generated based on Schnorr signature scheme proposed in [90]. Therefore, partial private key generation scheme using G_1 with hard DL assumption is provably secure in the random oracle.

3.5 Evaluation

In this section, experimental performance evaluation is done to evaluate the efficiency of the proposed scheme.

3.5.1 Experimental Setup

As an IoT device in the experiment, we used a smart phone (Samsung Galaxy Nexus with android version 4.2.2 with 11.88 GB storage size) as an average performance device because the smartphones are predicted to be a gateway of smart things in the mobile environment. ([91]-[93]) assume a similar environment.

For the existing schemes [78, 84, 85], we use a super singular elliptic curve over F_p having subgroup of order q where $p = 512$ bits, $q = 160$ bits, and length of $k0 = k = 192$ bits. For pairing operation, symmetric bilinear pairing $e : G_1 \times G_1 \to G_2$ is used where G_1 is an additive cyclic group of prime order q and G_2 is a multiplicative cyclic group of prime order q. Implementation is done using android library 5.1.1 and jpbc 1.2.1 library [94]. We choose three existing schemes for comparison, which are denoted as CME (Certificate-based Multi-receiver Encryption) [78], CLMS (Certificate-based Multi-receiver Signcryption) [84], and CLME (Certificate-less Multi-receiver Encryption) [85]. For the pro-

Table 3.3: Notations and Descriptions for Point Operation

Notation	Description
T_{pair}	Time cost for Pairing operation
T_{mul}	Time cost for Scalar multiplication
T_{pow}	Time cost for Exponentiation in G_1
T_{add}	Time cost for Addition in G_1
T_{expo}	Time cost for Exponentiation in G_2
T_{pdiv}	Time cost for Pairing division

posed scheme, to achieve the same security level, we use a non-singular elliptic curve over F_p having group of order q where $p = 160$ bits and $q = 160$ bits. To compare the computation cost, operation costs for elements in G_1 and G_2 are considered. Although the proposed scheme requires one more scalar multiplication for the generation of a public key $P_i^{''}$, it can be neglected as it is calculated just for once. In the experiment, the message payload size is set as 170 bytes.

3.5.2 Computational Cost Comparison

To compare the computational cost, the notations and descriptions for point operations in elliptic curve cryptography are shown in Table 3.3.

3.5.2.1 Encryption Cost

In the existing scheme CME, the encryption cost of the sender for $n-$receivers is $n(2T_{mul} + T_{add}) + T_{expo}$. For $n-$receivers, encryption time cost of sender by eliminating pre-computed pairing operation is $n(T_{mul} + T_{expo} + T_{pow}) + 2T_{mul}$ and $n(2T_{mul} + T_{add}) + 3T_{mul}$ for CLMS and CLME respectively. In CLMS, the encryption cost will be $n(2T_{mul} + T_{expo} + T_{pow} + T_{add}) + 2T_{mul}$ if the sender is not PKG that knows the master secret key. In the proposed scheme, $n(T_{mul} + T_{add}) + 2T_{mul}$ is required for $n-$receivers encryption. Computational cost for encryption is summarized in Table 3.4.

Experimental comparison of encryption time is shown in Fig. 3.3. In Fig. 3.3, the x-axis shows the number of receivers. For the corresponding receivers repre-

Table 3.4: Encryption Cost Comparison

	Encryption Cost for (n-receiver)			
	CME	CLMS	CLME	CLAME (Proposal)
T_{pair}	0	0 (or) 1	0 (or) 1	0
T_{mul}	2n	n+2	2n+3	n+2
T_{pow}	0	n	0	0
T_{add}	n	0	n	n
T_{expo}	1	n	0	0

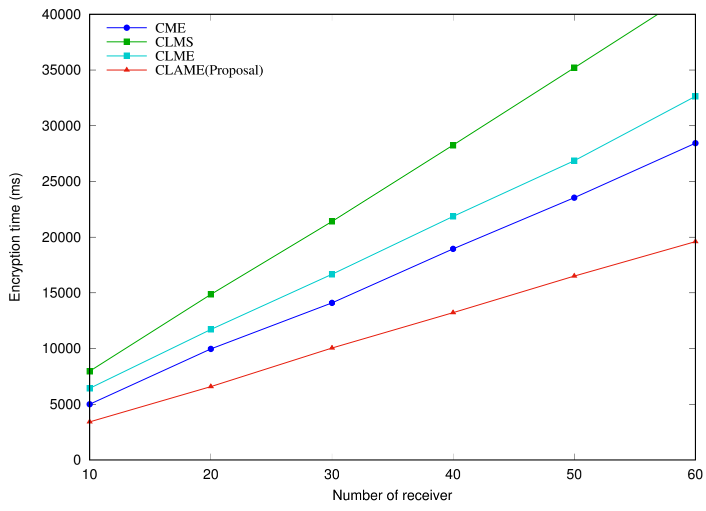


Figure 3.3: Experimental comparison of encryption time

sented by the x-axis value, the y-axis shows the results of the encryption time in milliseconds. Among existing schemes, CME requires less encryption time cost than that of CLMS and CLME. That is because the computational cost is less than that of CLMS and CLME. Although additional security properties are provided, the proposed scheme achieves faster encryption time for multi-receivers than that of all existing schemes because of less computational cost for encryption.

3.5.2.2 Decryption Cost

We consider decryption time cost for a receiver. CME requires $T_{pair} + T_{expo}$ for calculating σ' and $2T_{mul} + T_{add}$ for checking message integrity and U_i consistency.

Table 3.5: Decryption Cost Comparison

	Decryption Cost for a receiver			
	CME	CLMS	CLME	CLAME (Proposal)
T_{pair}	1	3	4	0
T_{mul}	2	2	0	4
T_{add}	1	2	0	4
T_{expo}	1	1	0	0
T_{pdiv}	0	0	1	0

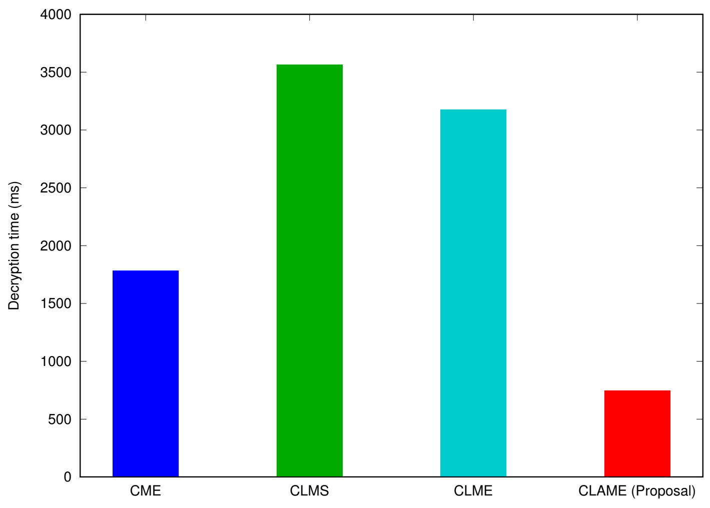


Figure 3.4: Experimental comparison of decryption time

The scheme CLMS requires $T_{pair} + T_{expo}$ for decryption and $2T_{pair} + 2T_{mul} + 2T_{add}$ for checking message authentication. In CLME, decryption cost is $4T_{pair}$ + T_{pdiv}. In the proposed scheme, decryption time cost is T_{mul} for calculating σ', T_{mul} for U_i consistency, and $2T_{mul} + 4T_{add}$ for checking message authentication and replay attack. Computational cost for decryption is compared in Table 3.5.

Experimental comparison of decryption time is shown in Fig. 3.4. The y-axis shows the result of decryption time on a receiver in milliseconds. Among all schemes, decryption cost of CLMS is the highest as it uses more computation expensive pairing operations. The decryption cost of CME is less than that of CLMS and CLME as it uses less pairing operation. The proposed scheme

Table 3.6: Security Properties Comparison

	CME	CLMS	CLME	CLAME(Proposal)
Key Escrow Problem	No	No	No	No
Message Integrity	Yes	Yes	Yes	Yes
Source Authentication	No	Yes	No	Yes
Replay Attack Prevention	No	No	No	Yes
Implicit User Authentication	No	Yes	Yes	Yes

(CLAME) achieves the lowest decryption cost because it avoids computation expensive pairing operations in decryption.

3.5.3 Security Properties Comparison

From the perspective of security properties, the schemes are summarized in Table 3.6. Although all schemes avoid key escrow problem, only proposed scheme provides replay attack prevention. The scheme described in CLMS and proposed scheme achieve source authentication. In the existing scheme CME, implicit user authentication property fails as the sender uses the public keys of receivers in encryption. Without checking the validity and authenticity of public keys, adversaries can easily impersonate as authorized receivers. To authenticate the receivers, the scheme will require two pairing operations.

3.6 Security Proofs

In this section, we show the proposed scheme is IND-CCA2 secure with confidentiality and provides authenticity, message integrity and replay attack prevention.

3.6.1 Confidentiality

For the confidentiality of the proposed scheme, security game is based on "Indistinguishability of encryptions under adaptive chosen ciphertext attacks" (IND-CCA2). The proposed scheme is (IND-CCA2) secure in the random oracle model if no polynomial time adversary $\mathcal{A}$ (Type-I or Type-II) has a non-negligible advantage in the following games played against the challenger C under hard ECDLP assumption.

3.6.1.1 Security Game for Type-I adversary

Theorem 1: When a polynomial time adversary $\mathcal{A}$ (Type-I adversary) can attack the scheme with advantage ϵ with the help of $H_i(1 \leq i \leq 4)$ random oracles, then there is an algorithm $\mathcal{B}$ that can solve ECDLP with non-negligible advantage. Let $hi-$list be the result of querying H_i random oracles respectively,where $(1 \leq i \leq 4)$.

Proof: Suppose that $\mathcal{B}$ has (P, bP) as an instance of the ECDLP.

- Setup: $\mathcal{B}$ runs the setup algorithm to generate public system parameters $params$ where $Pub_s = bP$ and master secret key $s \leftarrow \perp$. $\mathcal{B}$ gives $params$ to adversary $\mathcal{A}$.

- Phase 1: $\mathcal{B}$ outputs target identities $ID_j^* = \{ID_1^*, ID_2^*, ..., ID_n^*\}$, target sender ID_u and send them to $\mathcal{A}$. $\mathcal{B}$ randomly picks $d_u, sk_u \in Z_q$ randomly and computes $X_u = d_uP - h_uPub_s$, $P_u = sk_uP, P_u^{'} = P_u + X_u$ and $P_u^{''} = h_u^{-1}P_u^{'}$ where $h_u = H_1(ID_u, P_u, P_u^{'})$.

- Phase 2: $\mathcal{B}$ answers several queries with restricted adversary behavior $\mathcal{A}$. Assume L_p is the list of users that is initialized empty.

  1. CreateUser: If (ID_i, pk_i, sk_i, D_i) exists in L_p, $\mathcal{B}$ returns pk_i. Otherwise, it runs the following processes.

     (a) If $ID_i = ID_u$, then the challenger $\mathcal{B}$ sets $sk_u = \perp$ and $D_u = (X_u, d_u = \perp)$. It then returns $pk_u = (ID_u, P_u, P_u^{'}, P_u^{''})$ to $\mathcal{A}$.

     (b) If $ID_i \in ID_j^*$, $\mathcal{B}$ picks $d_i, sk_i \in Z_q$ randomly. Then, it computes $X_i = (d_i - h_i)Pub_s$, $P_i = sk_iPub_s, P_i^{'} = P_i + X_i$ and $P_i^{''} = h_i^{-1}P_i^{'}$ where $h_i = H_1(ID_i, P_i, P_i^{'})$. Then, it adds (ID_i, pk_i, sk_i, D_i) to L_p where $pk_i = (ID_i, P_i, P_i^{'}, P_i^{''})$ and $D_i = (X_i, d_i)$. Finally, it returns pk_i to adversary $\mathcal{A}$.

     (c) If $ID_i \notin ID_j^*$, $\mathcal{B}$ picks $d_i, sk_i \in Z_q$ randomly and computes $X_i = d_iP - h_iPub_s$, $P_i = sk_iP, P_i^{'} = P_i + X_i$ and $P_i^{''} = h_i^{-1}P_i^{'}$ where

$h_i = H_1(ID_i, P_i, P_i^{'})$. Then, it adds (ID_i, pk_i, sk_i, D_i) to L_p where $pk_i = (ID_i, P_i, P_i^{'}, P_i^{''})$ and $D_i = (X_i, d_i)$. Finally, it returns pk_i to $\mathcal{A}$.

2. Request-Public-Key: If (ID_i, pk_i, sk_i, D_i) exists in L_p, $\mathcal{B}$ returns the public key pk_i to $\mathcal{A}$. Otherwise, $\mathcal{B}$ makes CreateUser query.
3. Replace-Public-Key: If (ID_i, pk_i, sk_i, D_i) exists in L_p, $\mathcal{B}$ replaces the public key and set $sk_i = \perp$ and $d_i = \perp$.
4. Extract-Private-Key: If (ID_i, pk_i, sk_i, D_i) exists in L_p, C outputs sk_i. Otherwise, $\mathcal{B}$ runs CreateUser oracle. Then, it returns sk_i to $\mathcal{A}$.
5. Extract-Partial-Private-Key: If (ID_i, pk_i, sk_i, D_i) exists in L_p, $\mathcal{B}$ outputs D_i. Otherwise, $\mathcal{B}$ runs CreateUser oracle. Then, it returns D_i to $\mathcal{A}$.
6. Decrypt: Accepts (C^*, ID_i) as input where $< C^* = a^*, U_1, U_2, .., U_n, V^*, W^*, Y^*, L^*, t^* >$. If $(ID_i \in ID_j^*)$ or $(ID_i = ID_u)$ or t^* is not within reasonable time range, return "reject". Otherwise, the process is done as follows:
   (a) Perform Extract-Partial-Private-Key and Extract-Partial-Private oracles to get d_i, and sk_i.
   (b) Compute Z_i using U_i, d_i, and sk_i .
   (c) If $(Z_i, h2_i)$ exists in $h2-$ list, then compute $\sigma_i^{'} = V^* \oplus h2_i$. Otherwise, return "reject".
   (d) If $(\sigma_i^{'}, h4_i)$ exists in $h4-$list, then decrypt message $M^{'} = D_{h4_i}(W^*)$. Otherwise, return "reject".
   (e) If $(ID_i, P_i, P_i^{'}, h1_i)$ exists in $h1-$list and $(M^{'}||\sigma_i^{'}||L^*||t^*||Y^*, h3_i)$ in $h3-$list, then
      – check whether Y^* is equal to $a^*P - ((h3_i - h_u^{-1})P_u + P_u^{''} + Pub_s)$ or not. If it is equal, continue. Otherwise, return "reject".
      – check whether U_i and $h3_i h1_i(Pub_s + P_i^{''})$ are equal. If they are not equal, return "reject". Otherwise, return $M^{'}$ as the output plaintext.

- Challenge: $\mathcal{A}$ submits two plaintext messages (M_0, M_1) with the same length. $\mathcal{B}$ does the encryption by randomly selecting $\alpha \in \{0, 1\}$, $r \in Z_q$ and $\sigma \in$

$\{0,1\}^k$. It then computes $< C = a, U_1, U_2, .., U_n, V, W, Y, L, t >$ by performing the following steps and using the hash oracles

$$
\begin{aligned}
Y &= rP \\
a &= h_u^{-1} d_u + msk_u + r \, mod \, q \\
m &= H_3(M_\alpha \| \sigma \| L \| t \| Y) \\
Z &= mP \\
h_j &= H_1(ID_j, P_j, P_j^{'}) \\
U_j &= mh_j(Pub_s + P_j^{''}) \\
V &= \sigma \oplus H_2(Z) \\
K &= H_4(\sigma) \\
W &= E_K(M_\alpha)
\end{aligned}
$$

Type-I adversary's random oracles $H_i(1 \leq i \leq 4)$ work as follows:

- H_1-query: Accepts $(ID_i, P_i, P_i^{'})$ as input. If $(ID_i, P_i, P_i^{'}, h1_i)$ exists in $h1-$list, then $\mathcal{B}$ returns $h1_i$. Otherwise, do the following:
  1. Pick $h1_i \in Z_q$ randomly.
  2. Put $(ID_i, P_i, P_i^{'}, h1_i)$ in $h1-$list.
  3. Return $h1_i$.
- H_2-query: Accepts (Z_i) as input. If $(Z_i, h2_i)$ exists in $h2-$list, then $\mathcal{B}$ returns $h2_i$. Otherwise, do the following:
  1. Pick $h2_i \in \{0,1\}^{k0}$ randomly.
  2. Put $(Z_i, h2_i)$ in $h2-$list.
  3. Return $h2_i$.
- H_3-query: Accepts $(M_i \| \sigma_i \| L_i \| t_i \| Y_i)$ as input. If $(M_i \| \sigma_i \| L_i \| t_i \| Y_i, h3_i)$ exists in $h3-$list, then return $h3_i$. Otherwise, do the following:
  1. Pick $h3_i \in Z_q$ randomly.
  2. Put $(M_i \| \sigma_i \| L_i \| t_i \| Y_i, h3_i)$ in $h3-$list.
  3. Return $h3_i$.

  - H_4-query: Accepts (σ_i) as input. If $(\sigma_i, h4_i)$ exists in $h4-$list, then return $h4_i$. Otherwise, do the following:
    1. Pick $h4_i \in \{0,1\}^k$ randomly.
    2. Put $(\sigma_i, h4_i)$ in $h4-$list.
    3. Return $h4_i$.

- Guess: Finally, adversary $\mathcal{A}$ outputs $\alpha' \in \{0,1\}$. If $\alpha' = \alpha$, challenger $\mathcal{B}$ outputs 1 and $\mathcal{A}$ wins.

- Analysis: Type-I adversary $\mathcal{A}$ can break the IND-CCA2 security of the proposed scheme when $\mathcal{A}$ can find mP by computing $b^{-1}(d_j + sk_j)^{-1}U_j$ value. Therefore, $\mathcal{B}$ can solve ECDLP.

As discrete logarithm problem for finding b is computationally intractable in polynomial time, the proposed scheme is IND-CCA2 secure under hard DL assumption for the elliptic curve.

Suppose the Type-I adversary can guess the value of α with non-negligible advantage ϵ. If H_2 and H_3 are modeled as random oracles, $\mathcal{A}$ has advantage only if mP is the output of H_2 oracle or m is an output of H_3 oracle. The probability that the adversary can correctly guess the output of H_2 is $\frac{1}{2^{k0}}$. For q_{de} decryption queries, adversary has advantage $\epsilon - \frac{q_{de}}{2^{k0}}$. The probability that the adversary can correctly guess the output of H_3 is $\frac{1}{2^q}$. For q_{de} decryption queries, adversary has advantage $\epsilon - \frac{q_{de}}{2^q}$. Therefore, we know that the challenger $\mathcal{B}$ can address the ECDLP problem with non-negligible advantage $\epsilon - \frac{q_{de}}{2^{k0}}$ or $\epsilon - \frac{q_{de}}{2^q}$. As discrete logarithm problem is computationally intractable in polynomial time, the proposed scheme is IND-CCA2 secure against Type-I adversary $\mathcal{A}$.

3.6.1.2 Security Game for Type-II adversary

Theorem 2: When a polynomial time adversary $\mathcal{A}$ (Type-II adversary) can attack the scheme with advantage ϵ with the help of $H_i(1 \le i \le 4)$ random oracles, then there is an algorithm $\mathcal{B}$ that can solve ECDLP with non-negligible advantage. Let $hi-$list be the result of querying H_i random oracles respectively,where $(1 \le i \le 4)$.

Proof: Suppose that $\mathcal{B}$ has (P, bP) as an instance of the ECDLP.

- Setup: $\mathcal{B}$ runs the setup algorithm to generate public system parameters $params$ where $Pub_s = sP$ and master secret key s. $\mathcal{B}$ gives $params$ to adversary $\mathcal{A}$. Master secret key is also given to $\mathcal{A}$.

- Phase 1: $\mathcal{B}$ outputs target identities $ID_j^* = \{ID_1^*, ID_2^*, ..., ID_n^*\}$, target sender ID_u and send them to $\mathcal{A}$. $\mathcal{B}$ picks x_u, $sk_u \in Z_q$ randomly and computes $X_u = x_uP$, $P_u = sk_uP, P_u^{'} = P_u + X_u$ and $P_u^{''} = h_u^{-1}P_u^{'}$ where $h_u = H_1(ID_u, P_u, P_u^{'})$. Then, it calculates $d_u = h_u s + x_u \, mod \, q$.

- Phase 2: $\mathcal{B}$ answers several queries with restricted adversary behavior $\mathcal{A}$. Assume L_p is the list of users that is initialized empty.

  1. CreateUser: If (ID_i, pk_i, sk_i, D_i) exists in L_p, $\mathcal{B}$ returns pk_i. Otherwise, it runs the following processes.

     (a) If $ID_i = ID_u$, then the challenger $\mathcal{B}$ sets $sk_u = \perp$ and $D_u = (X_u, d_u = \perp)$. It then returns $pk_u = (ID_u, P_u, P_u^{'}, P_u^{''})$ to $\mathcal{A}$.

     (b) If $ID_i \in ID_j^*$, $\mathcal{B}$ picks $d_i, sk_i \in Z_q$ randomly. Then, it computes $X_i = d_i(bP)$, $P_i = sk_i(bP), P_i^{'} = P_i + X_i$ and $P_i^{''} = h_i^{-1}P_i^{'} - Pub_s$ where $h_i = H_1(ID_i, P_i, P_i^{'})$. And, it adds (ID_i, pk_i, sk_i, D_i) to L_p where $pk_i = (ID_i, P_i, P_i^{'}, P_i^{''})$ and $D_i = (X_i, d_i)$. Finally, it returns pk_i to adversary $\mathcal{A}$.

     (c) If $ID_i \notin ID_j^*$, $\mathcal{B}$ picks $x_i, sk_i \in Z_q$ randomly and computes $X_i = x_iP, P_i = sk_iP, P_i^{'} = P_i + X_i$ and $P_i^{''} = h_i^{-1}P_i^{'}$ where $h_i = H_1(ID_i, P_i, P_i^{'})$. Then, it calculates $d_i = h_i s + x_i \, mod \, q$ and adds (ID_i, pk_i, sk_i, D_i) to L_p where $pk_i = (ID_i, P_i, P_i^{'}, P_i^{''})$ and $D_i = (X_i, d_i)$. Finally, it returns pk_i to $\mathcal{A}$.

  2. Request-Public-Key: If (ID_i, pk_i, sk_i, D_i) exists in L_p, $\mathcal{B}$ returns the public key pk_i to $\mathcal{A}$. Otherwise, $\mathcal{B}$ makes CreateUser query.

  3. Replace-Public-Key: If (ID_i, pk_i, sk_i, D_i) exists in L_p, $\mathcal{B}$ replaces the public key and set $sk_i = \perp$ and $d_i = \perp$.

  4. Extract-Private-Key: If (ID_i, pk_i, sk_i, D_i) exists in L_p, C outputs sk_i. Otherwise, $\mathcal{B}$ runs CreateUser oracle. Then, it returns sk_i to $\mathcal{A}$.

5. Extract-Partial-Private-Key: If (ID_i, pk_i, sk_i, D_i) exists in L_p, $\mathcal{B}$ outputs D_i. Otherwise, $\mathcal{B}$ runs CreateUser oracle. Then, it returns D_i to $\mathcal{A}$.
6. Decrypt: Accepts (C^*, ID_i) as input where $< C^* = a^*, U_1, U_2, .., U_n, V^*, W^*, Y^*, L^*, t^* >$. If $(ID_i \in ID_j^*)$ or $(ID_i = ID_u)$or t^* is not within reasonable time range, return "reject". Otherwise, the process is done as follows:
   (a) Perform Extract-Partial-Private-Key and Extract-Partial-Private oracles to get d_i, and sk_i.
   (b) Compute Z_i using U_i, d_i, and sk_i .
   (c) If $(Z_i, h2_i)$ exists in $h2-$ list, then compute $\sigma_i^{'} = V^* \oplus h2_i$. Otherwise, return "reject".
   (d) If $(\sigma_i^{'}, h4_i)$ exists in $h4-$list, then decrypt message $M^{'} = D_{h4_i}(W^*)$. Otherwise, return "reject".
   (e) If $(ID_i, P_i, P_i^{'}, h1_i)$ exists in $h1-$list and $(M^{'}||\sigma_i^{'}||L^*||t^*||Y^*, h3_i)$ in $h3-$list, then
      – check whether Y^* is equal to $a^*P - ((h3_i - h_u^{-1})P_u + P_u^{''} + Pub_s)$ or not. If it is equal, continue. Otherwise, return "reject".
      – check whether U_i and $h3_i h1_i(Pub_s + P_i^{''})$ are equal. If they are not equal, return "reject". Otherwise, return $M^{'}$ as the output plaintext.

- Challenge: $\mathcal{A}$ submits two plaintext messages (M_0, M_1) with the same length. $\mathcal{B}$ does the encryption by randomly selecting $\alpha \in \{0, 1\}$, $r \in Z_q$ and $\sigma \in \{0, 1\}^k$. It then computes $< C = a, U_1, U_2, .., U_n, V, W, Y, L, t >$ by performing the following steps and using the hash oracles

$$\begin{aligned} Y &= rP \\ a &= h_u^{-1} d_u + msk_u + r \, mod \, q \\ m &= H_3(M_\alpha||\sigma||L||t||Y) \\ Z &= mP \\ h_j &= H_1(ID_j, P_j, P_j^{'}) \end{aligned}$$

$$U_j = mh_j(Pub_s + P_j^{''})$$
$$V = \sigma \oplus H_2(Z)$$
$$K = H_4(\sigma)$$
$$W = E_K(M_\alpha)$$

Type-II adversary's random oracles $H_i(1 \leq i \leq 4)$ work as follows:

- H_1−query: Accepts $(ID_i, P_i, P_i^{'})$ as input. If $(ID_i, P_i, P_i^{'}, h1_i)$ exists in $h1$−list, then $\mathcal{B}$ returns $h1_i$. Otherwise, do the following:
  1. Pick $h1_i \in Z_q$ randomly.
  2. Put $(ID_i, P_i, P_i^{'}, h1_i)$ in $h1$−list.
  3. Return $h1_i$.
- H_2−query: Accepts (Z_i) as input. If $(Z_i, h2_i)$ exists in $h2$−list, then $\mathcal{B}$ returns $h2_i$. Otherwise, do the following:
  1. Pick $h2_i \in \{0,1\}^{k0}$ randomly.
  2. Put $(Z_i, h2_i)$ in $h2$−list.
  3. Return $h2_i$.
- H_3−query: Accepts $(M_i||\sigma_i||L_i||t_i||Y_i)$ as input. If $(M_i||\sigma_i||L_i||t_i||Y_i, h3_i)$ exists in $h3$−list, then return $h3_i$. Otherwise, do the following:
  1. Pick $h3_i \in Z_q$ randomly.
  2. Put $(M_i||\sigma_i||L_i||t_i||Y_i, h3_i)$ in $h3$−list.
  3. Return $h3_i$.
- H_4−query: Accepts (σ_i) as input. If $(\sigma_i, h4_i)$ exists in $h4$−list, then return $h4_i$. Otherwise, do the following:
  1. Pick $h4_i \in \{0,1\}^k$ randomly.
  2. Put $(\sigma_i, h4_i)$ in $h4$−list.
  3. Return $h4_i$.

• Guess: Finally, adversary $\mathcal{A}$ outputs $\alpha^{'} \in \{0,1\}$. If $\alpha^{'} = \alpha$, challenger $\mathcal{B}$ outputs 1 and $\mathcal{A}$ wins.

- Analysis: Type-II adversary $\mathcal{A}$ can break the IND-CCA2 security of the proposed scheme when $\mathcal{A}$ can find mP by computing $b^{-1}(d_j + sk_j)^{-1}U_j$ value. Therefore, $\mathcal{B}$ can solve ECDLP.

As discrete logarithm problem for finding b is computationally intractable in polynomial time, the proposed scheme is IND-CCA2 secure under hard DL assumption for the elliptic curve.

Suppose the Type-II adversary can guess the value of α with non-negligible advantage ϵ. If H_2 and H_3 are modeled as random oracles, $\mathcal{A}$ has advantage only if mP is the output of H_2 oracle or m is an output of H_3 oracle. The probability that the adversary can correctly guess the output of H_2 is $\frac{1}{2^{k0}}$. For q_{de} decryption queries, adversary has advantage $\epsilon - \frac{q_{de}}{2^{k0}}$. The probability that the adversary can correctly guess the output of H_3 is $\frac{1}{2^q}$. For q_{de} decryption queries, adversary has advantage $\epsilon - \frac{q_{de}}{2^q}$. Therefore, we know that the challenger $\mathcal{B}$ can address the ECDLP problem with non-negligible advantage $\epsilon - \frac{q_{de}}{2^{k0}}$ or $\epsilon - \frac{q_{de}}{2^q}$. As discrete logarithm problem is computationally intractable in polynomial time, the proposed scheme is IND-CCA2 secure against Type-II adversary $\mathcal{A}$.

3.6.2 Source Authentication

Theorem 3: Under hard discrete logarithm assumption, the signature (a) is existentially unforgeable in the random oracle model.

Proof: Let $\mathcal{A}$ be any probabilistic time adversary with running time t_A, making q_s queries to signature oracle, and q_{H3} random oracle queries to H_3 oracle. $\mathcal{B}$ acts as a challenger and responds to $\mathcal{A}$'s signature and H_3 queries. Public key of third party, public key of target sender (pk_u) and X_u are given to challenger $\mathcal{B}$ as the signature public keys.

- Challenger $\mathcal{B}$ sends the signature public keys to $\mathcal{A}$. $\mathcal{B}$ then chooses $w \in [1, q_{H3}]$ randomly. Assume that the adversary $\mathcal{A}$ will forge the signature on the w-th H_3 query.

- signature oracle: For i-th signature query using message $(M_i, \sigma_i, L_i, t_i)$, $\mathcal{B}$ does the following

1. Choose random a_i, m_i from Z_q.
2. Set $Y_i = a_i P - m_i P_u - h_u^{-1}(h_u Pub_s + X_u)$.
3. return Y_i, a_i.
4. Store m_i as the value of $H_3(M_i \| \sigma_i \| L_i \| t_i \| Y_i)$.

- For j-th H_3 query on $(M_j \| \sigma_j \| L_j \| t_j \| Y_j)$, $\mathcal{B}$ does the following
  1. If the value of H_3 query already exists, return the value.
  2. If $j \neq w$, choose $m_j \leftarrow Z_q$ and set m_j as the result of H_3 query on $(M_j \| \sigma_j \| L_j \| t_j \| Y_j)$.
  3. If $j = w$, send Y_j to valid receiver of target sender and obtain a challenge m^* from that receiver. Then, hash value of $H_3(M_j \| \sigma_j \| L_j \| t_j \| Y_j)$ is set as m^*.
- On receiving a forgery attempt $(M^*, \sigma^*, L^*, t^*, Y^*, a^*)$ from $\mathcal{A}$, send a^* to valid receiver of target sender. If the j-th hash query is $(M^* \| \sigma^* \| L^* \| t^* \| Y^*)$ and $a^* P = Y^* + m^* P_u - h_u^{-1}(h_u Pub_s + X_u)$. Then, the signature forgery is valid and the source authentication property of the proposed scheme is broken.

The probability that the j-th hash query gets valid input will be $1/(q_{H3})$. And the probability that signature oracle issues duplicate Y value is $(q_{H3} + q_S + 1)/q$. Suppose the probability that adversary $\mathcal{A}$ can produce a valid signature forgery is ϵ. Then, there is an algorithm $\mathcal{B}$ that can impersonate the valid sender with probability at least $\epsilon/(q_{H3}) - (q_{H3} + q_S + 1)/q$.

To forge the valid signature a, the third party needs to reveal the value of sk_u from P_u. Suppose trusted third party calculates $h_u^{-1} d_u$ and $a - h_u^{-1} d_u$. Then, the result $msk_u + r$ is same to the Schnorr signature scheme. Therefore, the security of the source authentication lies on the ECDLP problem that is computationally intractable in polynomial time. As a result, the signature a is existentially unforgeable in the random oracle model.

3.6.3 Implicit User Authentication

Theorem 4: According to the security model for adversaries and the discrete logarithm assumption holds in the elliptic group, then it also provides implicit user authentication in the random oracle model with the restricted adversary behavior.

Proof: For implicit user authentication, the sender uses the public key of receivers and public key of trusted third party. Therefore, only receivers who have corresponding private key and master secret key of third party can decrypt the ciphertext. Receivers knows the master secret key of third party indirectly from valid partial private key. Among the oracles for different adversaries, the following two oracles also exist.

- Extract-Private-Key: This oracle accepts an identity as input and outputs the corresponding private key. The restriction of the oracle is that the adversary cannot request Extract-Private-Key if he or she has already replaced the public key for identity.
- Extract-Partial-Private-Key: This oracle takes an identity as input and returns the corresponding partial private key. The restriction of the oracle is that the adversary cannot request Extract-Partial-Private-Key if he or she has already replaced the public key for identity.

However, according to the security model, the Type-I adversary cannot access Extract-Partial-Private-Key and Extract-Private-Key random oracles for the target identities and target sender at any point. And the Type-II adversary cannot access Extract-Private-Key oracle for the target identities and target sender at any point. Therefore, finding the private key of receiver from its public key is the elliptic curve discrete problem. Therefore, the proposed scheme implicitly achieves user authentication.

3.6.4 Message Integrity

Theorem 5: For the message integrity, we consider the game played between polynomial time adversary $\mathcal{A}$ and challenger $\mathcal{B}$. The proposed scheme is secure against ciphertext forgery in the random oracle model if no polynomially-bounded adversary has a non-negligible advantage in the following game.

Proof:

Let $hi-$list be the result of querying H_i random oracles respectively,where $(1 \leq i \leq 4)$. The same hash random oracles as described in previous sections are used.

- Setup: $\mathcal{B}$ runs the setup algorithm to generate public system parameters *params* where $Pub_s = sP$ and master secret key s. $\mathcal{B}$ gives *params* to adversary $\mathcal{A}$.

- Phase 1: $\mathcal{B}$ outputs target identities $ID^*_j = \{ID^*_1, ID^*_2, ..., ID^*_n\}$, target sender ID_u and send them to $\mathcal{A}$.

- Phase 2: $\mathcal{B}$ answers several queries with restricted adversary behavior $\mathcal{A}$. Assume L_p is the list of users that is initialized empty.

  1. CreateUser: If (ID_i, pk_i, sk_i, D_i) exists in L_p, $\mathcal{B}$ returns pk_i. Otherwise, it runs the following processes.

     (a) If $ID_i = ID_u$ or $ID_i \in ID^*_j$, $\mathcal{B}$ picks $sk_i \in Z_q$ randomly. Then, it computes $X_i = bP$, $P_i = sk_iP, P'_i = P_i$ and $P''_i = h_i^{-1}P'_i$ where $h_i = H_1(ID_i, P_i, P'_i)$. Then, it assigns $sk_i = \perp$ and $d_i = \perp$. Then, it adds (ID_i, pk_i, sk_i, D_i) to L_p where $pk_i = (ID_i, P_i, P'_i, P''_i)$ and $D_i = (X_i, d_i)$. Finally, it returns pk_i to adversary $\mathcal{A}$.

     (b) If $ID_i \notin ID^*_j$, $\mathcal{B}$ picks $x_i, sk_i \in Z_q$ randomly and computes $X_i = x_iP$, $P_i = sk_iP, P'_i = P_i+X_i$ and $P''_i = h_i^{-1}P'_i$ where $h_i = H_1(ID_i, P_i, P'_i)$. Then, it performs $d_i = h_is+x_i \, mod \, q$. Then, it adds (ID_i, pk_i, sk_i, D_i) to L_p where $pk_i = (ID_i, P_i, P'_i, P''_i)$ and $D_i = (X_i, d_i)$. Finally, it returns pk_i to $\mathcal{A}$.

  2. Request-Public-Key: If (ID_i, pk_i, sk_i, D_i) exists in L_p, $\mathcal{B}$ returns the public key pk_i to $\mathcal{A}$. Otherwise, $\mathcal{B}$ makes CreateUser query.

  3. Replace-Public-Key: If (ID_i, pk_i, sk_i, D_i) exists in L_p, $\mathcal{B}$ replaces the public key and set $sk_i = \perp$ and $d_i = \perp$.

  4. Extract-Private-Key: If (ID_i, pk_i, sk_i, D_i) exists in L_p, C outputs sk_i. Otherwise, $\mathcal{B}$ runs CreateUser oracle. Then, it returns sk_i to $\mathcal{A}$.

  5. Extract-Partial-Private-Key: If (ID_i, pk_i, sk_i, D_i) exists in L_p, $\mathcal{B}$ outputs D_i. Otherwise, $\mathcal{B}$ runs CreateUser oracle. Then, it returns D_i to $\mathcal{A}$.

  6. Decrypt: Accepts (C^*, ID_i) as input where $< C^* = a^*, U_1, U_2, .., U_n, V^*, W^*, Y^*, L^*, t^* >$. If $(ID_i \in ID^*_j)$ or $(ID_i = ID_u)$ or t^* is not within rea-

sonable time range, return “reject”. Otherwise, the process is done as follows:

(a) Perform Extract-Partial-Private-Key and Extract-Partial-Private oracles to get d_i, and sk_i.

(b) Compute Z_i using U_i, d_i, and sk_i .

(c) If $(Z_i, h2_i)$ exists in $h2-$ list, then compute $\sigma_i^{'} = V^* \oplus h2_i$. Otherwise, return “reject”.

(d) If $(\sigma_i^{'}, h4_i)$ exists in $h4-$list, then decrypt message $M^{'} = D_{h4_i}(W^*)$. Otherwise, return “reject”.

(e) If $(ID_i, P_i, P_i^{'}, h1_i)$ exists in $h1-$list and $(M^{'}||\sigma_i^{'}||L^*||t^*||Y^*, h3_i)$ in $h3-$list, then

- check whether Y^* is equal to $a^*P-((h3_i - h_u^{-1})P_u + P_u^{''} + Pub_s)$ or not. If it is equal, continue. Otherwise, return “reject”.
- check whether U_i and $h3_i h1_i(Pub_s + P_i^{''})$ are equal. If they are not equal, return “reject”. Otherwise, return $M^{'}$ as the output plaintext.

- Challenge: Adversary $\mathcal{A}$ performs a number of queries described above to output a valid ciphertext from ID_u to ID_j^*.

- Guess: Finally, adversary $\mathcal{A}$ outputs ciphertext from ID_u to ID_j^*. If the ciphertext is valid, challenger $\mathcal{B}$ outputs 1 and $\mathcal{A}$ wins.

Suppose the adversary can forge the ciphertext with non-negligible advantage ϵ. If H_3 is modelled as random oracle, $\mathcal{A}$ has advantage only if m is an output of H_3 oracle. The probability that the adversary can correctly guess the output of H_3 is $\frac{1}{2^q}$. However, if signature unforgeability exists under hard discrete logarithm assumption, the proposed scheme is secure against the ciphertext forgery in the random oracle.

3.6.5 Replay attack prevention

In our proposed scheme, timestamp (t) is added in the calculation of m value. Again, (m) is used to create signature (a). Although timestamp value is publicly declared, replay attack is prevented as the signature a is existentially unforgeable in the random oracle model. Since it is secure against signature forgery in the random oracle model if no polynomially bounded adversary has non-negligible advantage, the proposed scheme is secure against replay attack in the random oracle model. Using timestamp for preventing replay attacks, synchronization of time is required. And the validity of timestamp should be restricted to a short time period in order to tolerate data delivery latency or time inaccuracy. Time synchronization can be done by using several time synchronization methods [101]. In the proposed scheme, we assume that semi-trusted party broadcasts time to achieve centralized time synchronization.

To avoid time synchronization, there are other ways such as challenge-response. Because both sender and receiver send nonce to each other for a unique value generation, challenge-response introduces more communication.

3.7 Conclusion

To reduce the computation load, we have proposed an efficient and secure multi-receiver encryption scheme with lightweight nature for the device to device communications of IoT applications. Our proposed scheme avoids the inherent key escrow problem as existing certificate-based and certificate-less multi-receiver encryption schemes do. Moreover, our proposed scheme achieves multi-receiver encryption with better efficiency and more security properties. Under Discrete Logarithm assumption, security proofs in the random oracles are also given for the proposed scheme. According to the computational cost comparison and experimental results, the proposed scheme achieves faster encryption and decryption time compared to existing schemes in multi-receiver environment.

Chapter 4

Design and Implementation of Reliable and Secure Multi-receiver Data Delivery System

4.1 Introduction

Another important issue in the multi-receiver data stream delivery is dynamic access policy changes. The data access policy may change depending on the context of the sender. Here, the data access policy means the policy defined by the data owner that grants who can access the data. For example, a patient shares his data with other patients having the same diseases or symptoms to get experiences and recommendations. When one of the receivers is intentionally stopped the subscription to the data stream delivery since the reputation of the receiver degrades (e.g. the receiver turned out to be a malicious or infected user), data access policy should be changed to prevent from that receiver. Such access policy changes often occur in IoT applications because of the feature of dynamicity and mobility. However, the problem of access policy changes in the multi-receiver stream delivery has not been studied enough in the existing multi-receiver stream delivery systems.

To solve the problem above and achieve reliable data analysis, we propose a reliable multi-receiver stream delivery system with encryption. Our proposed system adopts a certificate-less multi-receiver encryption scheme called CLAME (Certificate-Less Multi-receiver Encryption with Authentication) [97], which was proposed in Chapter 3, as a secret key sharing scheme for stream data encryption. In this chapter, we assume that data sharing is performed among the users. Data owner defines its target receiver group. Therefore, data sender is not a semi-trusted party, an honest but curious KGC. To achieve secure multi-receiver stream delivery, secret key needs to be shared among all receivers. Hereafter, we call the shared secret key as *shared secret*. The sender needs to run encrypt algorithm of CLAME scheme and uses encryption key as shared secret for future data delivery. Whenever the sender denies any one of its target receivers from data access, it is necessary to run encrypt algorithm of CLAME scheme again for shared secret key renewal. For reliable data delivery, our proposed system also adopts the SRSM scheme (Synchronized Recovery Stream Merging) [60], which was proposed in Chapter 2. However, directly adopting the SRSM scheme is inefficient because the load of the shared secret key renewal is not reduced when dealing with the access policy changes. Therefore, we propose an extension of the SRSM scheme to reduce the load of shared secret renewal process.

In the rest of this chapter, we first introduce the related work in Section 4.2. Then we present a system design that combines both the SRSM and CLAME schemes for multi-receiver stream delivery in Section 4.3. In Section 4.4, we propose an extension of the SRSM scheme that can reduce the load of shared secret renewal process, called the SRSM-R (Synchronized Recovery Stream Merging with a limited number of receivers) method. In Section 4.5, we evaluated the performance using a prototype implementation. Finally, we conclude this chapter in Section 4.6.

4.2 Related Work

For secure stream data delivery, Secure Multipurpose Internet Mail Extensions (S/MIME) provides necessary security properties. However, S/MIME is based on the asymmetric encryption. So, it requires a certificate for exchanging the public keys. Certificate-based multi-receiver encryption scheme has been proposed

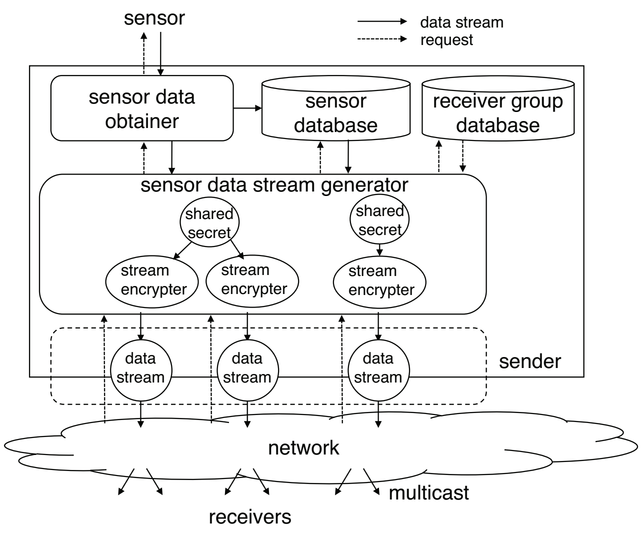


Figure 4.1: Proposal system model

in [78]. To avoid the overhead for certificate management, certificate-less multi-receiver encryption schemes have been proposed [84, 85]. Also, we have proposed a certificate-less multi-receiver encryption scheme with authentication (CLAME) for the device to device communications of Internet of Things (IoT) applications [97]. Proposed CLAME scheme avoids the inherent key escrow problem that existing certificate-based and certificate-less multi-receiver encryption schemes do. In the CLAME scheme, the data owner or the sender shares a secret key with the receivers. Using the shared secret key, the data are encrypted and delivered. And there is a possibility for the data owner to change the access control policy. When there are access right policy changes, shared secret renewal process is necessary to deliver the stream data encrypted by using multi-receiver encryption scheme. Because the renewing process of the shared secret is a kind of heavy computation load process, frequent access right policy changes can be a burden for the data stream sender.

To realize a reliable and secure multi-receiver stream delivery system, both reliable stream delivery scheme and encryption scheme need to be combined. However, such system design was not presented so far. In addition, the problem of shared secret renewal computation load has not been studied enough.

4.3 Proposal of the Multi-receiver Stream Delivery System

In this section we describe our design of the reliable and secure multi-receiver stream delivery system.

4.3.1 System model

Fig. 4.1 shows our proposed system model. The main difference from the system model of the SRSM to this model is that the generated data streams are encrypted ones. The stream generator includes stream encrypter, which encrypts the stream by shared secret. The shared secret is generated for each stream multicast. The same shared secret may be used by the multiple stream encrypters. If one of the receivers has left or joined, this shared secret key has to be generated again and distributed to all receivers. When the number of receivers becomes large, we have to renew shared secret key often. For example, a patient allows a number of other patients having the same diagnosis and symptoms to access his data. The patient shares a secret key with the receivers. When any one of the receivers misbehaved, then the patient renews the shared key to prevent the misbehaving receiver from future data access. In the case of large number of receivers, the chance of misbehaving receivers may increase. Since the proposed system allows the newly joined receiver to view the past stream data, the shared secret key renewal is needed when one of the receivers is stopped to get the data stream.

Therefore, to reduce the computation load in renewing shared secret, our proposal system splits the receivers into multiple groups and uses different shared secrets for them. By splitting the receivers into smaller groups, the shared secret key renewal computation load becomes smaller. However, the sender needs to provide multiple original streams or recovery streams for different groups. That is, there is a trade-off between the shared secret renewal computation load and the number of streams on the stream sender. Also, the system needs to manage which receiver belongs to which group. The receiver group database is used for managing the receiver groups that belong to the stream.

4.4 Proposal of reliable and secure multi-receiver stream delivery system

In this section, we describe the synchronized recovery stream merging with a limited number of receivers (SRSM-R) method and key renewal process.

4.4.1 SRSM-R

To implement a reliable and secure multi-receiver stream delivery system by simply combining the SRSM and CLAME schemes, we need to consider how to treat the computational overload problems for shared secret renewal process on the sender.

To generate shared secret for stream data encryption in recovery stream deliveries, we propose an extension of the SRSM scheme, which we call SRSM-R (Synchronized Recovery Stream Merging with a limited number of receivers) method. In the SRSM-R method, the number of receivers that can be handled by one original stream are limited. In other words, an original stream is assigned to a particular group of receivers. Multiple original streams are prepared to deliver data streams to multiple groups of the receivers. The number of groups is determined according to the parameter T, the threshold to generate a new group. If there are N receivers, then the number of groups G is represented as

$$G = \lceil \frac{N}{T} \rceil.$$

At this time, the number of receivers in one group is T or $N \bmod T$. If a receiver unsubscribed from the multicast delivery, the number of receivers in the corresponding group decreases. If a new receiver subscribes to the multicast delivery, the receiver is added as a member of a group having the number of receivers that is less than T. If all groups have T receivers, then a new group with one receiver is generated. Multi-receiver encryption is executed for each group. That means, all receivers in the same group use the same shared secret. The recovery stream generation and merging follow SRSM but the merging processes are independently executed for each original stream. The shared secret used for the recovery stream

Algorithm 3: Algorithm for the SRSM-R method

```
on receiving subscribe request from receiver $n$
begin
1      $g \leftarrow$ find_available_group();
2      if ($g = nil$) then
3          $g \leftarrow$ new_group($\{n\}$);
4          $g.secret \leftarrow$ new_shared_secret();
5          $s \leftarrow$ SRSM_new_stream();
6      else
7          $g \leftarrow g \cup \{n\}$;
8          $s \leftarrow$ SRSM_original_stream($g$);
       // notify $n$ the shared secret and stream identifier of $s$
9      unicast($n$, CLAME_encrypt($\{n\}$, $g.secret + s.id$));
on receiving unsubscribe request from receiver $n$
begin
10     $g \leftarrow$ find_group_contains($n$);
11     $g \leftarrow g - \{n\}$;
12     $g.secret \leftarrow$ new_shared_secret();
13     multicast($g$, CLAME_encrypt($g$, $g.secret$));
on receiving failure recovery request from receiver $n$
begin
14     $s \leftarrow nil$;
15     foreach $r \in$ SRSM_recovery_streams() do
16         if $r.size < T \wedge$ SRSM_mergeable_case_I($r, n$) then
17             $r.members \leftarrow r.members \cup n$;
18             $s \leftarrow r$;
19             $break$;
20     if $s = nil$ then
21         $s \leftarrow$ SRSM_generate_recovery_stream();
22         foreach $r \in$ SRSM_recovery_streams() do
23             if $r.size < T \wedge$ SRSM_mergeable_case_II($s, r$) then
24                 $s.merge(r)$;
       // notify $n$ the stream identifier of $s$
25     unicast($n$, AES_encrypt($g.secret$, $s.id$));
function delivery process on time $t$
begin
26     foreach $g \in groups$ do
27         foreach $s \in$ SRSM_all_streams($g$) do
28             multicast($s.members$, AES_encrypt($g.secret$, $s.data(t)$));
```

is the same as that of its original stream.

Algorithm 3 shows the pseudo code of the SRSM-R method describing the functions that are run on the sender. When a receiver requests to subscribe to a stream, the sender finds a group with available capacity so that the receiver can be added into it. If there is no such group, the sender has to create new group and add the receiver into it. For secure data delivery, a new shared secret is generated for that group. The sender then sends the newly generated shared secret and the corresponding stream identifier encrypted by CLAME as the response. If there is a group with available capacity at the receiver's stream subscription time, the receiver is assigned to that group on the sender. And the corresponding shared secret and stream identifier are sent to the receiver by unicast with CLAME encryption. The receiver then joins the multicast with the corresponding stream identifier. The stream identifier which is sent to the receiver corresponds to the original stream for that group. The newly joined receiver can decrypt the stream data delivered in the past. Conditional statements starting from line number (1) to (9) handle subscribe request from the receiver.

When a receiver unsubscribed from a group, the sender finds the group to which the receiver belongs. Then the receiver is removed from that group and the shared secret renewal process is executed. And the newly generated shared secret is delivered to the rest of the group members by multicast with CLAME encryption. Conditional statements from line number (10) to (13) describe the function for unsubscribe request.

When a recovery request is received from a receiver, the SRSM process is executed keeping the member size threshold T. After SRSM stream handling, new recovery stream identifier is delivered to the receiver with AES encryption. Conditional statements from line number (14) to (25) describe the above situation. Then the sender delivers the stream data by multicast with AES encryption using corresponding shared secret key for every time slot. Conditional statements for data delivery process at time t are shown from line number (26) to (28).

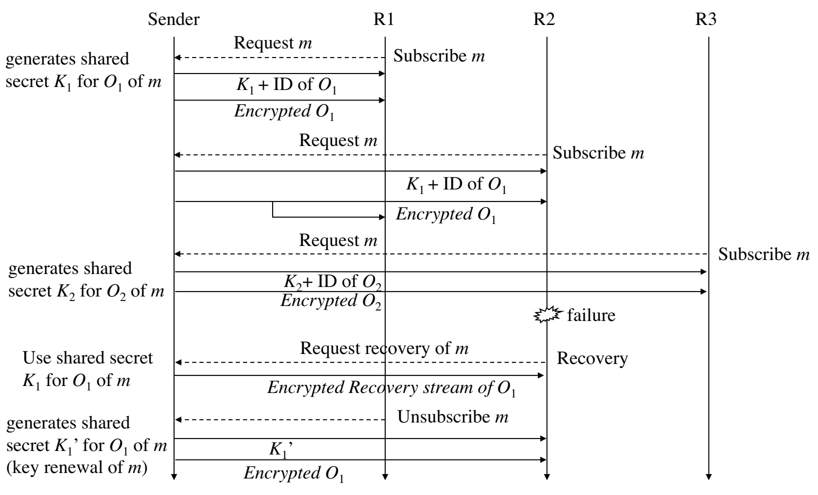


Figure 4.2: A communication sequence

4.4.2 Shared secret renewal in SRSM-R

In the SRSM-R method, the sender needs to modify the access control policy whenever any receiving member no longer wants the data or any one of the receivers was expelled from data access rights. In other words, the sender has to conduct the shared secret renewal process. In every shared secret renewal process, the sender generates new shared secret, encrypts the data stream with it and prepares the parameters for the receivers who meet the access control policy. The parameters are prepared to hide the new shared secret so that only intended receivers can reveal it.

Fig. 4.2 shows an example sequence of the communication for delivering a multicast data stream which has identifier m in our system. In this figure, three receivers ($R1$, $R2$ and $R3$) subscribe to m. For simple explanation, we assume the threshold $T = 2$. As the value of $N = 3$ and $T = 2$, the sender splits the encrypted data stream into two groups. In this example, as a response to subscription request of m by $R1$ and $R2$, an original stream O_1 is generated. When $R3$ requests subscription to m, a new original stream O_2 is generated. Different shared secrets are assigned for each original stream. In this case, K_1 for O_1 (the receivers are $R1$ and $R2$) and $K2$ for O_2 (the receiver is $R3$). The sender sends two multicast data streams using corresponding shared secret. When the receiver $R2$ encounters

Table 4.1: Simulation setup

Parameter	Value
Recovery speed	2.0
Acceptable latency	200 (unit time)
Simulation length	1,000 (unit time)
Number of tests	1,000

data loss, $R2$ sends a recovery request to the sender. Upon receiving the recovery request, the sender delivers a recovery stream. The same shared secret with the original stream is used for the recovery stream. In this case, K_1 is used to encrypt the recovery data stream for $R2$. When a receiver unsubscribed from the data stream multicast, shared secret renewal process is executed. In this example, $R1$ sends unsubscribe request of m and the members of O_1 is changed. Then the sender generates a new shared secret K_1', encrypts the stream data for O_1 with it, and delivers the encrypted data to the remaining receivers, in this case, $R2$.

4.5 Implementation and Performance Evaluation

In this section, we show some evaluation results of our secure multicast data delivery system. Our evaluation is based on the measurement using an implementation of CLAME scheme and the simulation of the SRSM-R method.

4.5.1 Experimental Setup

To evaluate the performance and feasibility of the proposals, we implemented a security mechanism that follows our system design. The experiment uses the same key size, the same implementation parameters and the same comparison schemes that are used in Chapter 3. The proposed scheme is implemented by using bouncy castle library [1], encryption library in JAVA. The evaluation program runs on nexus 7 tablet having android version 4.4.2 with Quad-core 1.2 GHz Cortex-A9 CPU and 1 GB RAM. Assuming the stream data size is small, in the experiment, the

[1]http://www.bouncycastle.org/

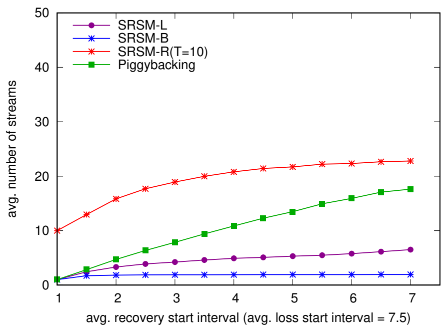


Figure 4.3: Average number of streams (T=10)

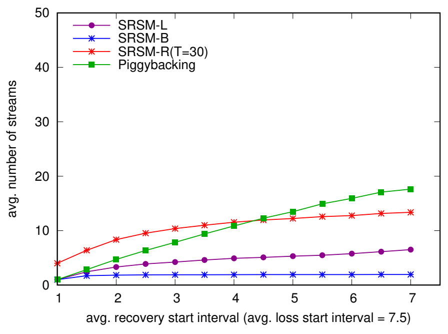


Figure 4.4: Average number of streams (T=30)

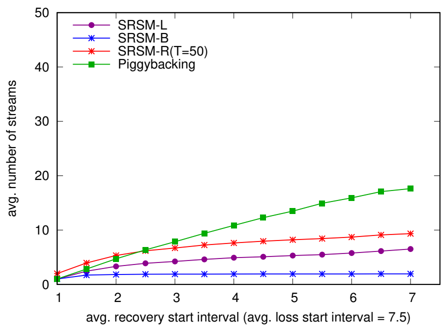


Figure 4.5: Average number of streams (T=50)

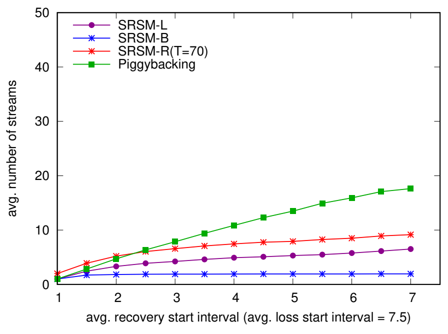


Figure 4.6: Average number of streams (T=70)

message payload size is set as 256 bits, which corresponds to the typical size of the shared secret.

The simulation setup is shown in Table 4.1. To see the basic performance, all receivers are assumed to have the same requirements for acceptable stream speed and acceptable latency in the simulations. The bandwidth of the recovery stream is twice as that of the original stream, which corresponds to the situation in which half of the stream data is skipped or skip rate is 2.0. The acceptable latency is set as 200 unit time. The result is the average of 1,000 times simulation. The simulation interval is set as 1,000 unit time, which is enough duration for the stable simulation and to see the behavior of the scheme.

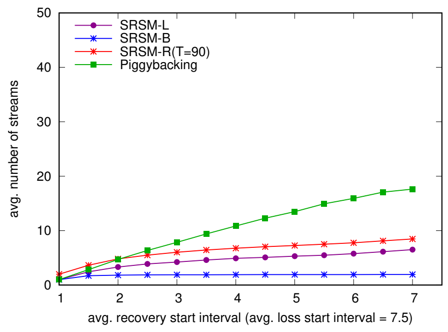


Figure 4.7: Average number of streams (T=90)

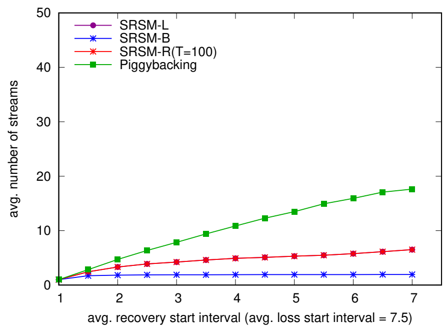


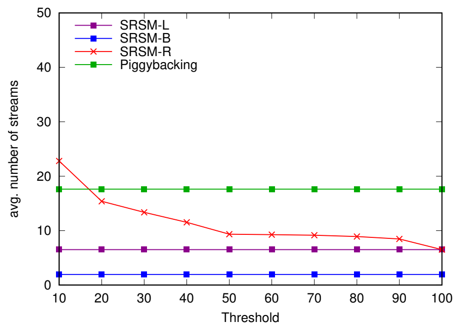
Figure 4.8: Average number of streams (T=100)

4.5.2 Performance Evaluation

Firstly, the number of streams managed by the sender is evaluated by simulations. The failure and recovery of the receivers follow the Poisson process.

Fig. 4.3 to Fig. 4.8 show the average number of streams comparison for the SRSM-L, SRSM-B, SRSM-R methods using odd number threshold values that are between 10 and 100, and the Piggybacking. Since the SRSM-L method, the SRSM-B method and the Piggybacking do not consider the grouping of the receivers, the average number of streams is constant when the threshold value changes. And the results of average number of streams in the SRSM-L and SRSM-B methods are comparatively smaller than that of the SRSM-R method and the Piggybacking. From the evaluation results, we can see that the average number of streams in the SRSM-R method is inversely proportional to the threshold value. When the threshold value becomes larger, then the average number of streams becomes smaller. The performance of the SRSM-R method outperforms the Piggybacking in the case of larger threshold value with longer recovery start interval. Since the total number of receivers are 100, there will be only one group when the SRSM-R method uses the threshold value 100. As a result, the SRSM-L method and the SRSM-R method with threshold value 100 require the same average number of streams. As explained in Chapter 2, the SRSM-B method achieves the best performance among all schemes.

Using the fixed recovery interval, the comparison of the average number of streams is summarized in Fig. 4.9. In Fig. 4.9, the y-axis shows the average num-



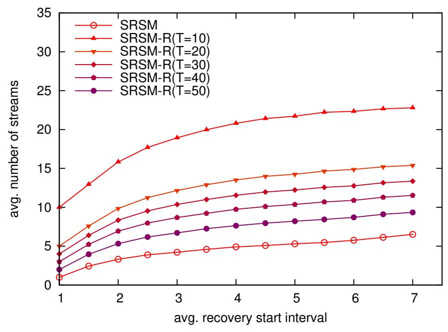


Figure 4.9: Average number of streams (Fixed recovery interval)

Figure 4.10: Average number of streams (Changed recovery interval)

ber of streams through the simulation time and the x-axis shows the threshold value. In the simulation, the average recovery start interval is 7 while average loss start interval is set as 7.5. The average number of streams for the SRSM-L method, the SRSM-B method, and the Piggybacking is constant for all threshold values. In the SRSM-R method, the average number of stream decreases when the threshold value increases.

Changing the average recovery start interval while keeping average loss start interval the same, Fig. 4.10 shows the result of the average number of streams generated for the SRSM-L and SRSM-R methods, changing the threshold T from 10 to 50. The larger the average recovery start interval is, the larger the loss interval becomes. In Fig. 4.10, the x-axis shows the the average failure recovery start interval.

Then, we evaluated the cost of shared secret renewal by using time metric in our implementation.

Fig. 4.11 to Fig. 4.14 show shared secret renewal time cost measured in seconds for the combination of the SRSM-R method and different multi-receiver encryption schemes, including existing encryption schemes and CLAME. Existing encryption schemes are denoted as CME (Certificate-based Multi-receiver Encryption) [78], CLMS (Certificate-less Multi-receiver Signcryption [84], and CLME (Certificate-less Multi-receiver Encryption) [85]. The x-axis represents the threshold value ranging from 10 to 100 and y-axis represents the shared secret renewal time. Since the SRSM-L method, the SRSM-B method and the Piggy-

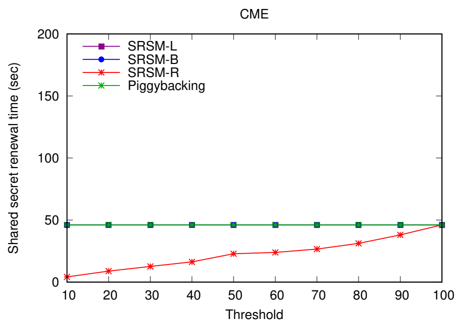


Figure 4.11: Shared secret renewal time comparison using CME

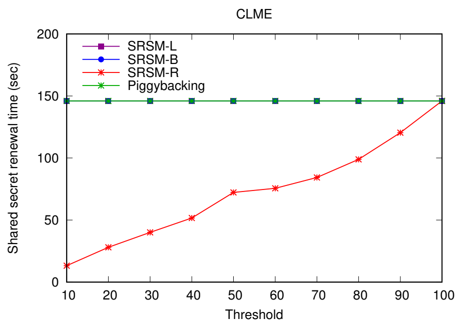


Figure 4.12: Shared secret renewal time comparison using CLME

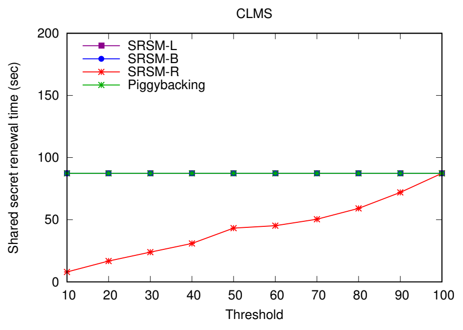


Figure 4.13: Shared secret renewal time comparison using CLMS

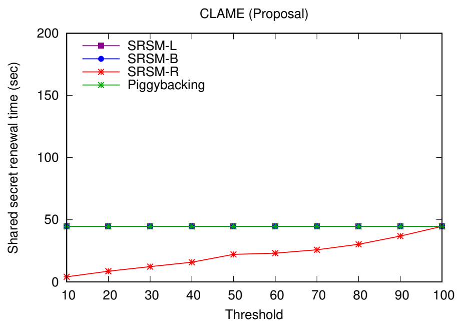


Figure 4.14: Shared secret renewal time comparison using CLAME (proposal)

backing do not divide the receivers into groups, the shared secret renewal costs are the same for all different multi-receiver encryption schemes. Among all encryption schemes, the proposal scheme (CLAME) achieves the best efficiency for shared secret renewal time.

Fig. 4.15 shows the summary of shared secret renewal cost comparison using the combination of the SRSM-R method and different multi-receiver encryption schemes. Fig. 4.15 also shows a result of shared secret renewal time using combination of SRSM (SRSM or SRSM-R) and CLAME. In the graph, the y-axis shows the shared secret renewal time. The x-axis shows threshold T. We changed T from 10 to 100. The result shows that the CLAME takes the shortest time for shared secret renewal among the encryption schemes. In addition, we can say the integration of CLAME and the SRSM-R method requires shorter shared secret renewal time for all T values. Though the shared secret renewal time of the

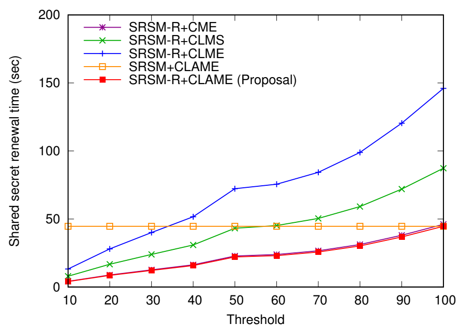


Figure 4.15: Shared secret renewal time comparison

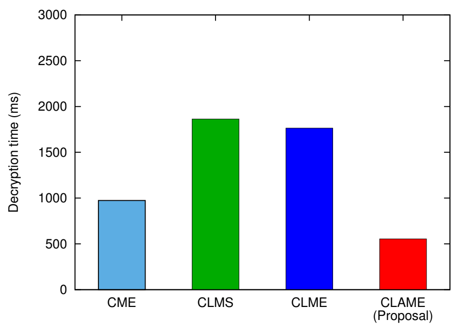


Figure 4.16: Experimental comparison of decryption time

proposed scheme and CME is almost equal, the decryption time of the proposed scheme is shorter as described in Fig. 4.16.

In summary, there is a trade-off between the key renewal cost and the number of streams in the proposed method that combines CLAME and the SRSM-R method. Appropriate T value may differ depending on the computational ability and available network bandwidth of the sender. For example, if the T is set as 50 while the number of receivers is 100, about 2.8 streams increase on average when the average recovery start interval is 7.0. If the sender has plenty of network bandwidth, but limits on CPU, smaller T is appropriate. If the sender has limited network bandwidth but has plenty of CPU power, larger T is appropriate. Smaller T value brings less key renewal time. According to the report [2], a security blocking software blocked approximately 550 cyber attacks each day on an Android device. That is, each device is attacked every 157 seconds. If 10 % of the devices do not have such blocking software, they may be infected every 15.7 seconds. That means we should quarantine those devices as misbehaved users. Therefore, we can estimate that a leave occurs every 15.7 seconds on average because of misbehavior when there are 100 devices as in the simulation setup in Table 4.1. It is the worst case but the system needs to be tolerant for such cases. In that case, we can choose $T = 30$ where our proposed method can achieve about 10 seconds for a key renewal time as shown in Fig. 4.14. To achieve a similar key

[2]https://www.symantec.com/content/dam/symantec/docs/security-center/archives/istr-16-april-volume-21-en.pdf

renewal time, other methods require smaller T value. For example, as shown in Fig. 4.12, CLME has to set $T = 15$. In this case, as shown in Fig. 4.9, the SRSM-R method needs twice number of streams in some fixed recovery interval which requires more network bandwidth. The reason why the SRSM-R method needs twice the number of streams is that the receivers are classified into some groups and the sender has to prepare the streams for each group. In the case of $T = 15$ in the simulation, the receivers are classified into seven groups and the sender has to prepare 7 streams though this is just 1 stream under the SRSM-L method, which is the smallest value. Moreover, the sender eliminates the chance of stream merging that can exist among receivers from different groups. The SRSM-R method is better in the point that we can control the key renewal cost and the number of streams by changing the parameter T. However, how to choose optimal value of T is more complicated and we set it as a future work. Since a smaller T value brings a larger number of recovery streams as shown in Fig. 4.9. Therefore, we consume more network bandwidth. The balance between a key renewal cost and a network bandwidth is a difficult design decision.

4.6 Conclusion

In this chapter, we proposed a novel reliable and secure multi-receiver stream delivery system. We propose a signaling protocol that covers the reliable multicast stream delivery scheme (SRSM) and a certificate-less multi-receiver encryption scheme with authentication (CLAME). To cope with the overheads of access right policy changes on CLAME, we proposed the SRSM-R method, an extension of SRSM that divides the receivers into groups to limit the number of receivers who uses the same shared secret for the encryption. By implementing the scheme on an Android terminal, we evaluated the performance to see the feasibility and effectiveness of our proposals. In addition we evaluated the performance of the SRSM-R method by simulations on probabilistic model. We confirmed the trade-off between the shared secret renewal cost and the number of streams.

Chapter 5

Conclusion

5.1 Summary

In multi-receiver data delivery for IoT applications, we have discussed two issues that affect the reliable data analysis and data sensitivity. In this thesis, we mainly focus on the solutions to these issues. In Chapter 1, we presented the importance of sensor data reliability and security, the necessity of multi-receiver data delivery, and then discussed research issues.

In Chapter 2, we proposed a scheme called synchronized recovery stream merging (SRSM) for sensor data multicasting. In order to cope with the network bandwidth limitations of the sender's devices, the SRSM scheme synchronizes and merges multiple recovery streams. We presented two methods of the SRSM scheme, latency-aware synchronized recovery stream merging (SRSM-L) method and bandwidth-dependent synchronized recovery stream merging method called the SRSM-B method. In the SRSM-L method, the receivers specify tolerable latency or waiting time. The idea of the SRSM-L method is that the receivers who received more data have to wait other recovery streams for some time so that stream merging can occur. In the SRSM-B method, stream merging emphasizes the available network bandwidth of the sender. We evaluated our proposed methods for two failure/recovery simulation scenarios. And, in the frequent random failure situation, the results show that the network bandwidth requirement of the SRSM-L method for multi-receiver data delivery is small compared with the existing schemes. In the burst failure situation, the SRSM-B method achieves the best performance.

In Chapter 3, we focused on the security issue of multi-receiver data delivery. In IoT applications, the sender and the receivers often have resource-constrained devices. In secure multi-receiver data delivery, the sender's load for encryption increases when the number of receivers increases. And the less decryption cost is preferable to the receivers. Therefore, we proposed a lightweight multi-receiver encryption that is suitable for IoT environments. The proposed scheme reduces the computational loads for encryption and decryption by avoiding the computation expensive operations. Besides, the necessary security requirements for multi-receiver data delivery such as confidentiality, message authentication, and replay attacks are fulfilled. The experimental results have shown that the proposed scheme reduces the encryption and decryption time cost, and also achieves more security properties.

In Chapter 4, we proposed a method called synchronized recovery stream merging with a limited number of receivers (SRSM-R) method, which divides the receivers into some groups where each group consists of the receivers that are handled by an original stream. For reliable and secure multi-receiver data delivery, we combined the synchronized recovery stream merging methods and the encryption scheme proposed in Chapter 3. However, there is another issue when any one of the receivers left the data delivery. The issue is the renewal of the shared secret key. In the SRSM-R method, each receiver belongs to a group and key renewal is performed separately. Through the simulation, we confirmed that less key renewal cost is achieved in the SRSM-R method by using smaller threshold value although there is a trade-off between the key renewal time and the average number of streams.

In summary, this thesis shows that our proposed synchronized recovery stream merging methods reduce the computational and the communication costs for multi-receiver data delivery on Internet of Things (IoT). In addition, our proposed solution for security issues reduces the computational cost by preserving the necessary security properties. Moreover, our proposed methods reduce the key renewal time cost while providing reliable and secure multi-receiver data delivery.

5.2 Future Work

The future work can be divided into four parts.

The first future challenge for this work is dynamically selecting the appropriate stream merging method for multicast data delivery according to the data loss feedbacks. In real scenarios, the available bandwidth of the sender changes along with time. Moreover, the contexts of the receivers, which means the resource availability, desired data quality and communication overhead of the receivers, may differ. Therefore, data delivery scheme should consider the dynamic selection of appropriate stream merging method to get a better performance for the receivers.

The second one is merging streams among different groups. In the SRSM-R method, we only consider stream merging within the same group. By merging streams among groups, the possibility to merge more streams increases. In case where the sender has a limit on its network bandwidth, the bandwidth consumption can be reduced by merging more streams. To establish this, we focus on stream merging among different groups.

From the security aspect, revocable certificate-less multi-receiver encryption is our future work. In Chapter 4, if we combine broadcast encryption [99], we could find efficient way for revocation and the shared key renewal. In large IoT applications such as healthcare, instead of sender-defined access policy, any misbehaving user should be revoked so that other remaining users can avoid interaction with that misbehaving user. Moreover, there may be cases where the attackers get the private key of the user or the reputation of the user degrades. These lead insecurity for the remaining users to interact with a malicious user. In such cases, the misbehaving users should be revoked.

Finally, in the evaluation, we just simulated the communication environment. To get more realistic evaluation, our last future work is the evaluation of the proposed methods in the real network environments.

Acknowledgement

Upon the completion of this thesis, I would like to take this opportunity to express my sincerest gratitude to those who have done their best to support me.

First and foremost, I would like to express my sincere gratitude to my supervisor, Professor Shinji Shimojo, who not only accepted me as his student but also assisted me to overcome the limitation of my knowledge as well as providing me valuable advice and guidance throughout my doctoral research work.

In addition, I would like to express my appreciation to Associate Professor Tomoki Yoshihisa at Osaka University, Assistant Professor Tomoya Kawakami at Nara Institute of Science and Technology, Yoshimasa Ishi at Osaka University and Dr. Yuuichi Teranishi at National Institute of Information and Communications Technology. This thesis would have never finished without their valuable advice and guidance.

I am also grateful to Professor Makoto Onizuka and Professor Toru Fujiwara at Graduate School of Information Science and Technology, Department of Multimedia Engineering, Osaka University, for their valuable time giving useful comments to improve the quality of this thesis.

My sincere thanks also goes to Professor Takahiro Hara and Professor Matsushita Yasuyuki of the Department of Multimedia Engineering, the Graduate School of Information Science and Technology at Osaka University for their appropriate guidance.

Last but not least, I would like to offer my hearty thanks to my family who always appreciate my every decision, share my good or bad times, and provide me with their greatest concerns and cares.

Finally, I thank all who have given me strong mental support throughout my study in Japan, especially my friends and kind-hearted people at Shimojo Laboratory.

REFERENCE

[1] Khalil, N., Abid, M.R., Benhaddou, D., and Gerndt, M.: Wireless sensors networks for Internet of things. In *Proc. of 2014 IEEE 9th Intl. Conf. on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pp. 1–6, 2014.

[2] Flammini, A. and Sisinni, E.: Wireless sensor networking in the Internet of things and Cloud Computing Era. In *Procedia Engineering J.*, Elsevier, Vol. 87, pp. 672–679, 2014.

[3] Chi, Q., Yan, H., Zhang, C., Pang, Z., and Da Xu, L.: A reconfigurable smart sensor interface for industrial WSN in IoT environment. In *IEEE Trans. on industrial informatics*, Vol. 10, No. 2, pp. 1417–1425, 2014.

[4] Hromic, H., Le Phuoc, D., Serrano, M., Antonic, A., Zarko, I.P., Hayes, C., and Decker, S.: Real time analysis of sensor data for the Internet of things by means of clustering and event processing. In *Proc. of 2015 IEEE Intl. Conf. on Communications (ICC)*, pp. 685–691, 2015.

[5] Islam, T., Mukhopadhyay, S.C., and Suryadevara, N.K.: Smart sensors and Internet of things: A postgraduate paper. In *IEEE Sensors J.*, Vol. 17, No. 3, pp. 577–584, 2017.

[6] Castillo, A. and Thierer, A.D.: Projecting the growth and economic impact of the Internet of things. In Mercatus Center at George Mason University, Economic Persvectives, 10 pages, 2015.

[7] Sun, Y., Song, H., Jara, A.J., and Bie, R.: Internet of things and big data analytics for smart and connected communities. In *IEEE Access J.*, Vol. 4, pp. 766–773, 2016.

[8] Suraki, M.Y. and Jahanshahi, M.: Internet of things and its benefits to improve service delivery in public health approach. In *Proc. of 2013 7th Intl. Conf. on Application of Information and Communication Technologies (AICT)*, pp. 1–4, 2013.

[9] Formisano, C., Pavia, D., Gurgen, L., Yonezawa, T., Galache, J.A., Doguchi, K., and Matranga, I.: The advantages of IoT and cloud applied to smart cities. In *Proc. of 2015 3rd Intl. Conf. on Future Internet of Things and Cloud (FiCloud)*, Vol. 4, pp. 325–332, 2015.

[10] Jimenez, F. and Torres, R.: Building an IoT-aware healthcare monitoring system. In *Proc. of 2015 IEEE 34th Intl. Conf. on the Chilean Computer Science Society (SCCC)*, pp. 1–4, 2015.

[11] Khan, S.F.: Health care monitoring system in Internet of things (IoT) by using RFID. In *Proc. of 2017 IEEE 6th Intl. Conf. on Industrial Technology and Management (ICITM)*, pp. 198–204, 2017.

[12] Gomez, J., Oviedo, B., and Zhuma, E.: Patient monitoring system based on Internet of things. In *Procedia Computer Science J.*, Elsevier, Vol. 83, pp. 90–97, 2016.

[13] Li, S.: Application of the Internet of things technology in Precision Agriculture IRrigation Systems. In *Proc. of 2012 Intl. Conf. on Computer Science and Service System*, pp. 1009–1013, 2012.

[14] Dlodlo, N. and Kalezhi, J.: The internet of things in agriculture for sustainable rural development. In *Proc. of 2015 Intl. Conf. on Emerging Trends in Networks and Computer Communications (ETNCC)*, pp. 13–18, 2015.

[15] Abdelgawad, A. and Yelamarthi, K.: Structural health monitoring: Internet of things application. In *Proc. of 2016 IEEE 59th Intl. Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 1–4, 2016.

[16] Arcadius, T.C., Gao, B., Tian, G., and Yan, Y.: Structural health monitoring framework based on Internet of things: A Survey. In *J. IEEE Internet of Things*, Vol. 4, No. 3, pp. 619–635, 2017.

[17] Xiu, Z. and Li, H.: Smart lighting system with brightness and color temperature tunable. In *Proc. of 2014 7th Intl. Symposium on Computational Intelligence and Design (ISCID)*, Vol. 2, pp. 183–186, 2014.

[18] Abinaya, B., Gurupriya, S., and Pooja, M.: IoT based smart and adaptive lighting in street lights. In *Proc. of 2017 2nd Intl. Conf. on Computing and Communications Technologies (ICCCT)*, pp. 195–198, 2017.

[19] Ouerhani, N., Pazos, N., Aeberli, M., and Muller, M.: IoT-based dynamic street light control for smart cities usecases. In *Proc. of 2016 Intl. Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1–5, 2016.

[20] Khanna, A. and Anand, R.: IoT-based smart parking system. In *Proc. of Intl. Conf. on Internet of Things and Applications (IOTA)*, pp. 266–270, 2016.

[21] Lee, C., Han, Y., Jeon, S., Seo, D., and Jung, I.: Smart parking system for Internet of things. In *Proc. of Intl. Conf. on Consumer Electronics (ICCE)*, pp. 263–264, 2016.

[22] Tsaramirsis, G., Karamitsos, I., and Apostolopoulos, C.: Smart parking: An IoT application for smart city. In *Proc. of 3rd Intl. Conf. on Computing for Sustainable Global Development (INDIACom)*, pp. 1412–1416, 2016.

[23] Roy, A., Siddiquee, J., Datta, A., Poddar, P., Ganguly, G., and Bhattacharjee, A.: Smart traffic and parking management using IoT. In *Proc. of 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 1–3, 2016.

[24] Patru, I.I., Carabas, M., Barbulescu, M., and Gheorghe, L.: Smart home IoT system. In *Proc. of 2016 15th RoEduNet Conference: Networking in Education and Research*, pp. 1–6, 2016.

[25] Verma, H., Jain, M., Goel, K.,Vikram, A., and Verma, G.: Smart home system based on Internet of Things. In *Proc. of 2016 3rd Intl. Conf. on Computing for Sustainable Global Development (INDIACom)*, pp. 2073–2075, 2016.

[26] Manna, S., Bhunia, S.S., and Mukherjee, N.: Vehicular pollution monitoring using IoT. In *Proc. of Intl. Conf. on Recent Advances and Innovations in Engineering (ICRAIE2014)*, pp. 1–5, 2014.

[27] Xiaojun, C., Xianpeng, L., and Peng, X.: IOT-based air pollution monitoring and forecasting system. In *Proc. of 2015 Intl. Conf. on Computer and Computational Sciences (ICCCS)*, pp. 257–260, 2015.

[28] Spalazzi, L., Taccari, G., and Bernardini, A.: An Internet of things ontology for earthquake emergency evaluation and response. In *Proc. of 2014 Intl. Conf. on Collaboration Technologies and Systems (CTS)*, pp. 528–534, 2014.

[29] Alphonsa, A. and Ravi, G.: Earthquake early warning system by IOT using Wireless sensor networks. In *Proc. of Intl. Conf. on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pp. 1201–1205, 2016.

[30] Diot, C., Levine, B.N., Lyles, B., Kassem, H., and Balensiefen, D.: Deployment issues for the IP multicast service and architecture. In *IEEE Network*, Vol. 14, No. 1, pp. 78–88, 2000.

[31] Banerjee, S. and Bhattacharjee, B.: Scalable secure group communication over IP multicast. In *IEEE J. on Selected Areas in Communications*, Vol. 20, No. 8, pp. 1511–1527, 2002.

[32] Ratnasamy, S., Ermolinskiy, A., and Shenker, S.: Revisiting IP multicast. In *Proc. of 2006 Conf. on Applications, technologies, architectures, and protocols for computer communications*, Vol. 36, No. 4, pp. 15–26, 2006.

[33] Banerjee, S., Bhattacharjee, B., and Kommareddy, C.: Scalable application layer multicast. In *Proc. of 2002 Conf. on Applications, technologies, architectures, and protocols for computer communications*, Vol. 32, No. 4, pp. 205–217, 2002.

[34] Hosseini, M., Ahmed, D.T., Shirmohammadi, S., and Georganas, N.D.: Survey of application-layer multicast protocols. In *IEEE Communications Surveys and Tutorials*, Vol. 9, No. 3, pp. 58–74, 2007.

[35] Hunkeler, U., Truong, H.L, Stanford-Clark, A.: MQTT-S — A publish/subscribe protocol for wireless sensor networks. In *3rd Intl. Conf. on Communication Systems Software and Middleware and Workshops*, pp. 791–798, 2008.

[36] Vinoski, S.: Advanced message queuing protocol. In *IEEE Internet Computing*, Vol. 10, No. 6, pp. 87–89, 2006.

[37] Bormann, C., Castellani, A.P., and Shelby, Z.: CoAP: An Application Protocol for Billions of Tiny Internet Nodes. In *IEEE Internet Computing*, Vol. 16, No. 2, pp. 62–67, 2012.

[38] Patel, M., Naughton, B., Chan, C., Sprecher, N., Abeta, S., and Neal, A.: Mobile-edge computing - Introductory technical White Paper. In *MEC industry initiative*, pp. 1–36, 2014.

[39] Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L.: Edge computing: Vision and challenges. In *IEEE Internet of Things J.* , Vol. 3, No. 5, pp. 637–646, 2016.

[40] Porambage, P., Braeken, A., Schmitt, C., Gurtov, A., Ylianttila, M., and Stiller, B.: Group key establishment for enabling secure multicast communication in wireless sensor networks deployed for IoT applications. In *IEEE Access*, Vol. 3, pp. 1503–1511, 2015.

[41] Jiang, D., Xu, Z., and Lv, Z.: A multicast delivery approach with minimum energy consumption for wireless multi-hop networks. In *Telecommunication systems*, Springer, Vol. 62, No. 4, pp. 771–782 , 2016.

[42] Quinn, B. and Almeroth, K.: IP multicast applications: Challenges and solutions. In *RFC 3170*, 2001.

[43] Eugster, P.T., Felber, P.A., Guerraoui, R., and Kermarrec, A.M.: The many faces of publish/subscribe. In *ACM computing surveys (CSUR) J.*, Vol. 35, No. 2, pp. 114–131, 2003.

[44] MQTT Version 3.1.1: `https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.pdf`

[45] Banno, R., Takeuchi, S., Takemoto, M., Kawano, T., Kambayashi, T., and Matsuo, M.: Designing overlay networks for handling exhaust data in a distributed topic-based Pub/Sub architecture. In *J. of Information Processing*, IPSJ, Vol. 23, No. 2, pp. 105–116, 2015.

[46] Teranishi, Y., Banno, R., and Akiyama., T.: Scalable and locality-aware distributed topic-based pub/sub messaging for IoT. In *Proc. of IEEE GLOBECOM 2015*, pp. 1–7, 2015.

[47] Verbelen, V., Simoens, P., De Turck, F. and Dhoedt, B.: Cloudlets: Bringing the cloud to the mobile user. In *Proc. of 3rd ACM Workshop on Mobile cloud computing and services*, pp. 29–36, 2012.

[48] Dan, A., Shahabuddin, P., Sitaram, D., and Towsley, D.: Channel allocation under batching and VCR control in video-on-demand systems. In *J. of Parallel and Distributed Computing*, Elsevier, Vol. 30, No. 2, pp. 168–179, 1995.

[49] Golubchik, L., Lui, J., and Muntz, R.: Reducing I/O demand in video-on-demand storage senders. In *SIGMETRICS Performance Evaluation Review*, ACM, Vol. 23, No. 1, pp. 25–36, 1995.

[50] Aggarwal, C., Wolf, J., and Yu, P.S.: On optimal piggyback merging policies for Video-on-Demand systems. In *Proc. of ACM Intl. Conf Measurement and Modeling of Computer Systems*, Vol. 24, No. 1, pp. 200–209, 1996.

[51] Aggarwal, C.C., Wolf, J.L., and Yu, P.S.: On optimal batching policies for video-on-demand storage servers. In *Proc. of 3rd IEEE Intl. Conf. on Multimedia Computing and Systems*, pp. 253–258, 1996.

[52] Golubchik, L., Lui, J.C., and Muntz.R. R.: Adaptive piggybacking: A novel technique for data sharing in video-on-demand storage servers. In *Multimedia Systems J.*, pp. 140–155, 1996.

[53] Carter, S.W. and Long, D.D.: Improving video-on-demand server efficiency through stream tapping. In *Proc. of 6th Intl. Conf. on Computer Communications and Networks*, IEEE, pp. 200–207, 1997.

[54] Hua, K.A., Cai, Y., and Sheu, S.: Patching: A multicast technique for true video-on-demand services. In *Proc. of 6th ACM Intl. Conf. on Multimedia*, ACM, pp. 191–200, 1998.

[55] Eager, D. and Vernon, M.: Dynamic skyscraper broadcasts for video-on-demand. In *Advances in Multimedia Information Systems (MIS 1998). LNCS*, Springer, Vol. 1508, pp. 18–32, 1998.

[56] Lau, S.W., Lui, J.C., and Golubchik, L.: Merging video streams in a multimedia storage server: complexity and heuristics. In *Multimedia Systems J.*, Springer, Vol. 6, No. 1, pp. 29–42, 1998.

[57] Carter, S.W. and Long, D.D.: Improving bandwidth efficiency of video-on-demand servers. In *Computer Networks J.*, Elsevier, Vol. 31, No. 1, pp. 111–123, 1999.

[58] Coffman Jr, E.G., Jelenkovic, P., and Momcilovic, P.: Provably efficient stream merging. In *Web Caching and Content Delivery*, pp. 171–184, 2001.

[59] Bar-Noy, A., Goshi, J., Ladner, R.E., and Tam, K.: Comparison of stream merging algorithms for media-on-demand. In *Multimedia Systems J.*, Springer, Vol. 9, No. 5, pp. 411–423, 2004.

[60] Teranishi, Y., Ei Khaing Win, Yoshihisa, T., and Shimojo, S.: A sensor data stream recovery scheme for event-driven IoT applications. In *Proc. of IEEE GLOBECOM2017*, pp. 1–6, 2017.

[61] McKay, K.A., Bassham, L., Turan, M.S., and Mouha, N.: Report on lightweight cryptography. In *NIST DRAFT NISTIR J.*, Vol. 8114, 2016.

[62] Dhillon, P.K. and Kalra, S.: Elliptic curve cryptography for real time embedded systems in IoT networks. In *2016 5th Intl. Conf. on Wireless Networks and Embedded Systems (WECON)*, pp. 1–6, IEEE, 2016.

[63] Gutmann, P.: PKI: it's not dead, just resting. In *Computer J.* , Vol. 35, No. 8, pp. 41–49, 2002.

[64] Shamir, A.: Identity-based cryptosystems and signature schemes. In *Proc. of Advances in Cryptology - CRYPTO'84. LNCS*, Springer, Vol. 196, pp. 47–53, 1984.

[65] Al-Riyami, S.S. and Paterson, K.G.: Certificateless public key cryptography. In *Proc. of Intl. Conf. on the Theory and Application of Cryptology and Information Security (ASIACRYPT). LNCS*, Springer, Vol. 2894, pp. 452–473, 2003.

[66] Baek, J., Safavi-Naini, R., and Susilo, W.: Efficient multi-receiver identity-based encryption and its application to broadcast encryption. In *Proc. of Intl. Workshop on Public Key Cryptography (PKC). LNCS*, Springer, Vol. 3386, pp. 380–397, 2005.

[67] Delerablée, C.: Identity-based broadcast encryption with constant size ciphertexts and private keys. In *Proc. of Intl. Conf. on the Theory and Application of Cryptology and Information Security (ASIACRYPT). LNCS*, Springer, Vol. 4833, pp. 200–215, 2007.

[68] Sakai, R. and Furukawa, J.: Identity-based broadcast encryption. In *IACR Cryptology ePrint Archive J.*, Vol. 2007, pp. 217–230, 2007.

[69] Jiang, H., Xu, Q., and Shang, J.: An efficient dynamic identity-based broadcast encryption scheme. In *Proc. of 2010 2nd Intl. Symposium on Data, Privacy and E-Commerce (ISDPE)*, IEEE, pp. 27–32, 2010.

[70] Hu, L., Liu, Z., and Cheng, X.: Efficient identity-based broadcast encryption without random oracles. In *J. of Computers*, Vol. 5, No. 3, pp. 331–336, 2010.

[71] Ming, Y. and Shen, X.: Multi-receiver Identity-based key encapsulation in the standard model. In *Proc. of 2010 Intl. Conf. Information Science and Management Engineering*, IEEE, Vol. 1, pp. 382–385, 2010.

[72] Fan, C.I., Huang, L.Y., and Ho, P.H.: Anonymous multireceiver identity-based encryption. In *Trans. Computers*, IEEE, Vol. 59, No. 9, pp. 1239–1249, 2010.

[73] Zhang, J. and Mao, J.: An improved anonymous multi-receiver identity-based encryption scheme. In *J. of Communication Systems*, Wiley Online Library, Vol. 28, No. 4, pp. 645–658, 2015.

[74] Fan, C.I. and Tseng, Y.F.: Anonymous multi-receiver identity-based authenticated encryption with CCA security. In *Symmetry J.*, Multidisciplinary Digital Publishing Institute, Vol. 7, No. 4, pp. 1856–1881, 2015.

[75] Wang, S.: Practical identity-based encryption (IBE) in multiple PKG environments and its applications. In *arXiv preprint cs/0703106 J.*, 2007.

[76] Qin, L., Cao, Z., and Dong, X.: Multi-receiver identity-based encryption in multiple PKG environment. In *Proc. of IEEE GLOBECOM2008*, IEEE, pp. 1–5, 2008.

[77] Selvi, S.S.D, Vivek, S.S., Srinivasan, R., and Rangan, C.P.: An efficient identity-based signcryption scheme for multiple receivers. In *Proc. of Intl. Workshop on Security. Advances in Information and Computer Security (IWSEC). LNCS*, Springer, Vol. 5824, pp. 71–88, 2009.

[78] Sur, C., Jung, C.D., and Rhee, K.H.: Multi-receiver certificate-based encryption and application to public key broadcast encryption. In *Proc. of 2007 ECSIS Symposium on Bio-inspired, Learning, and Intelligent Systems for Security (BLISS2007)*, IEEE, pp. 35–40, 2007.

[79] Lee, Y.R. and Lee, H.S.: An authenticated certificateless public key encryption scheme. In *Trends in Mathematics Information Center for Mathematical Sciences*, Vol. 8, No. 1, pp. 177–187, 2005.

[80] Baek, J., Safavi-Naini, R., and Susilo, W.: Certificate-less public key encryption without pairing. In *Proc. of Intl. Conf. on Information Security (ISC). LNCS*, Springer, Vol. 3650, pp. 134–148, 2005.

[81] Sun, Y., Zhang, F., and Baek, J.: Strongly secure certificateless public key encryption without pairing. In *Proc. of Intl. Conf. on Cryptology and Network Security (CANS). LNCS*, Springer, Vol. 4856, pp. 194–208, 2007.

[82] Choi, K.Y., Park, J.H., Hwang, J.Y., and Lee, D.H.: Efficient certificateless signature schemes. In *Applied Cryptography and Network Security. LNCS*, Springer, Vol. 4521, pp. 443–458, 2007.

[83] Selvi, S.S.D., Vivek, S.S., Shukla, D., and Rangan, C.P.: Efficient and provably secure certificateless multi-receiver signcryption. In *Proc. of Intl. Conf. on Provable Security (ProvSec). LNCS*, Springer, Vol. 5324, pp. 52–67, 2008.

[84] Selvi, S.S.D., Vivek, S.S, and Rangan, C.P.: A note on the Certificateless Multi-receiver Signcryption Scheme. In *IACR Cryptology ePrint Archive, Report*, Vol. 2009, pp. 308–316, 2009.

[85] Zhu, J., Chen, L.L,, Zhu, X., and Xie, L.: A new efficient certificateless multi-receiver public key encryption scheme. In *Intl. J. of Computer Science Issues (IJCSI)*, Vol. 13, No. 6, pp. 1–7, 2016.

[86] S.Miller, V.: Uses of elliptic curves in cryptography. In *Proc. of Advances in Cryptology — CRYPTO '85. LNCS*, Springer, pp. 417–426, 1985.

[87] Koblitz, N.: Elliptic curve cryptosystems. In *Mathematics of Computation*, Vol. 48, No. 177, pp. 203–209, 1987.

[88] Menezes, A.: An Introduction to Pairing-based Cryptography. In *Mathematics Subject Classification, Primary 94A60*, 1991.

[89] Jansma, N. and Arrendondo, B.: Performance comparison of elliptic curve and rsa digital signatures. In *nicj. net/files J.*, 2004.

[90] Schnorr, C.P.: Efficient identification and signatures for smart cards. In *Advances in Cryptology - CRYPTO'89. LNCS*, Springer, Vol. 435, pp. 239—252, 1989.

[91] Pereira, C., Rodrigues, J., Pinto, A., Rocha, P., Santiago, F., Sousa, J., and Aguiar, A.: Smartphones as M2M gateways in smart cities IoT applications. In *Proc. of 2016 23rd Intl. Conf. on Telecommunications (ICT)*, IEEE, pp. 1–7, 2016.

[92] Piyare, R. and Lee, S.R.: Smart home-control and monitoring system using smart phone. In *Proc. of 1st Intl. Conf. on Convergence and its Application (ICCA)*, Vol. 24, pp. 83–86, 2013.

[93] Dimitrov, D.V.: Medical internet of things and big data in healthcare. In *Healthcare Informatics Research J.*, Vol. 22, No. 3, pp. 156–163, 2016.

[94] De Caro, A. and Iovino, V.: jPBC: Java pairing based cryptography. In *Proc. of 16th IEEE Symposium on Computers and Communications (ISCC)*, IEEE, pp. 850–855, 2011.

[95] Teranishi, Y., Kimata, T., Yamanaka, H., Kawai, E., and Harai, H.: Dynamic data flow processing in edge computing environments. In *Proc. of Intl. Conf. on IEEE Computer Software and Applications 2017 (COMPSAC2017)*, pp. 935–944, 2017.

[96] Ei Khaing Win, Yoshihisa, T., Ishi, Y., Kawakami, T., Teranishi, Y., and Shimojo, S.: A lost sensor data recovery scheme for sensor data stream multicasting. In *J. of Information Processing*, Vol. 26, 2018 (to appear).

[97] Ei Khaing Win, Yoshihisa, T., Yoshimasa, I., Kawakami, T., Teranishi, Y., and Shimojo, S.: A lightweight multi-receiver encryption scheme with mutual authentication. In *Proc. of IEEE Intl. COMPSAC Workshop on Security, Trust and Privacy for Software Applications (STPSA'17)*, pp. 491–497, 2017.

[98] Selent, D.: Advanced Encryption Standard. In *RIVIER ACADEMIC J.*, pp. 1–14, 2010.

[99] Fiat, A. and Naor, M.: Broadcast encryption. In *Annual Intl. Cryptology Conf.* , Springer, Berlin, Heidelberg, pp. 480–491, 1993.

[100] Watanabe, Y., Shikata, J., and Imai, H.: Equivalence between semantic security and indistinguishability against chosen ciphertext attacks. In *Intl. Workshop on Public Key Cryptography*, Springer, Berlin, Heidelberg, pp. 71–84, 2003.

[101] Sivrikaya, F. and Yener, B.: Time synchronization in sensor networks: a survey. In *IEEE network*, Vol. 18, No. 4, pp. 45–50, 2004.