Vol.2017-DPS-172 No.5
Vol.2017-SPT-26 No.5
Vol.2017-EIP-78 No.5
2017/11/30

Design and Implementation of a Reliable and Secure Multi-receiver Stream Delivery System

Ei Khaing Win[1,a)] Yuuichi Teranishi[1,2] Yoshimasa Ishi[1] Tomoya Kawakami[3,1] Tomoki Yoshihisa[1] Shinji Shimojo[1]

Abstract: In this paper, we propose a novel reliable and secure multi-receiver stream delivery system. Our proposal system applies a reliable multicast stream delivery scheme called SRSM (Synchronized Recovery Stream Merging), which efficiently resends streams for loss-encountered receivers to achieve reliable data analysis. For secure data delivery in dynamic environment, we applied a certificate-less multi-receiver encryption scheme called CLAME (Certificate-Less Authenticated Multi-receiver Encryption). To alleviate the CPU overloads caused by shared secret key renewal when there are access right policy changes, we also propose an extension of SRSM called SRSM-R (Synchronized Recovery Stream Merging with Receiver number limitations). We evaluated the performance using a prototype implementation on Android device. We confirmed that the shared secret key renewal time can be reduced by 50% if the sender can accept extra 2.8 streams on average for threshold value 50.

1. Introduction

In the emerging Internet of Things (IoT) technologies, network-connected sensors or sensor-attached devices are used for sensing purposes. The sensor data streams generated continuously by a sensor are utilized for various objectives in IoT applications. For example, a video data stream obtained by a camera can be used for suspect tracking, congestion detection, situation logging, weather analysis, disaster prevention, and so on. In the edge computing environment [1], [2], in which many computers run on the network edges to realize quick-response on IoT applications, a large number of edge computer devices must receive a sensor data stream to analyze the situations of the real world [3]. In such applications, instead of one-to-one communication, one-to-many communication, i.e. multicasting, is an efficient means to deliver the same data to multiple receivers.

There are two issues that need to be addressed in multicasting for practical IoT applications. The first issue is data losses. The users or devices may encounter data loss for several reasons such as unexpected power on/off of the devices, unreliable network links or underground places with no network availability. In case of data loss, to keep the reliability of the result of sensor data analysis, retransmitting the lost data to the receivers is necessary. The sensor data sender that stores the archived sensor data needs to retransmit data to the loss-encountered receivers. Since the data losses can occur asynchronously, basically the sender needs to generate new channel to send the lost data, which becomes the burden of computational or network load. In the IoT applications in which embedded devices or mobile devices can be a data sender, energy efficiency is important to keep the battery life longer. Therefore, reducing the number of streams for the lost data recovery is an important issue.

The second issue is data sensitivity. Many IoT applications require to treat personal sensing data such as locations by localization sensors, activities by gyroscopes or heartbeat sensors, and so on. In the proprietary IoT applications, collected data are not allowed to disclose to others. Such sensor data are required to be protected from leakages to the unintended receivers or malicious attackers. An important issue to be treated 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 such as location or time. For example, data access policy relies on the location of the patient who want to share his own data with nearby doctors. The data access policy may also change because of the number of receivers (join or leave). For example, when a receiver intentionally stops the subscription to the data stream delivery or the reputation of a receiver decreases (e.g. the receiver turned out to be an malicious or infected user), access of future data needs to be prevented from that receiver. Such access policy changes can occur frequently in the IoT applications because of its nature of dynamicity and mobility.

However, existing multi-receiver stream delivery systems do not treat both reliability and security properties. In addition, the problem of access policy changes in the multi-receiver stream delivery has not been studied enough.

In this paper, we propose a reliable multi-receiver stream delivery system with encryption that treats such issues. The system assumes to deliver a reliable stream using SRSM (Synchronized Recovery Stream Merging) [4], which efficiently resends streams for multiple loss-encountered receivers for event-driven data analysis. The system applies a certificate-less multi-receiver encryption scheme called CLAME (Certificate-Less Authenticated Multi-receiver Encryption) [5] as a secret key sharing scheme for stream data encryption. To achieve secure multi-receiver stream

[1] Osaka University, Suita, Osaka, Japan
[2] NICT, Koganei, Tokyo, Japan
[3] NAIST, Ikoma, Nara, Japan
[a)] ei.khaing.win@ais.cmc.osaka-u.ac.jp

delivery, secret key needs to be shared among all receivers. Hereafter, we call the shared secret key as *shared secret*. CLAME is a lightweight encryption scheme for multi-receiver data delivery. It has security features for checking message integrity, source authentication, replay attack prevention and implicit user authentication.

The contribution of the paper is two-folds. First, we present a system design that combines both SRSM and CLAME for multi-receiver stream delivery. Second, we propose an extension of SRSM that can reduce the load of shared secret renewal process for encryption, called SRSM-R (Synchronized Recovery Stream Merging with Receiver number limitations), for frequent access policy changes. We evaluated the performance using a prototype implementation on Android smartphone device and simulations to show the feasibility and effectiveness of our proposals.

2. Related Work

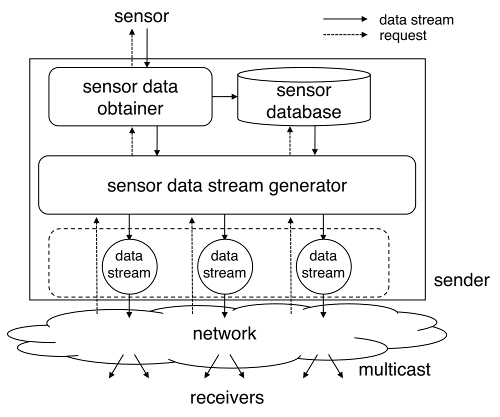
To reduce the number of data streams for asynchronous requests and alleviate network bandwidth usage on the sender, there are many studies that treat stream merging schemes in the literature. In these schemes, the sender or the receiver adjusts the process rate or delivery speed to merge into the original stream. Patching [6], Tapping [7], Dynamic Skyscraper [8], Piggybacking [9] etc. are well-known schemes. These schemes assume Video-on-Demand(VoD) applications. The latency of data processing does not matter in the VoD applications because the data are stored data and valid as long as the video is displayed on the receivers smoothly. Therefore, latency is not required to be preserved in these merging schemes. In addition, most of them assume that the receiver has plenty of memories or storage as a buffer to save delivered streams so that the jitter of the display process is kept as small as possible. These properties in the previous works do not match the requirement of IoT applications. Therefore, we proposed a stream merging scheme called SRSM, that does not require plenty of buffer on the receiver side and can reduce the number of streams to alleviate the load of sender [4].

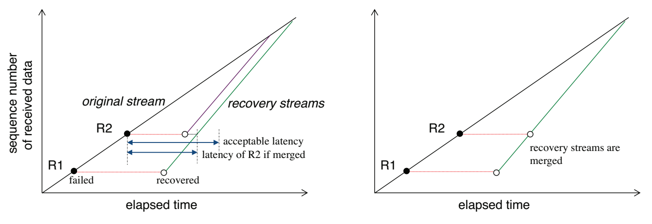
As multi-receiver encryption scheme, many encryption schemes have been proposed [10], [11], [12]. We have proposed an efficient and secure multi-receiver encryption scheme with lightweight nature for the device to device communications of Internet of Things (IoT) applications which we call CLAME [5]. CLAME avoids the inherent key escrow problems that existing certificate-based and certificateless multi-receiver encryption schemes do. When there are access right policy changes, shared secret renewal process is necessary for stream data multicast encryption using multi-receiver encryption schemes. 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 stream data 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.

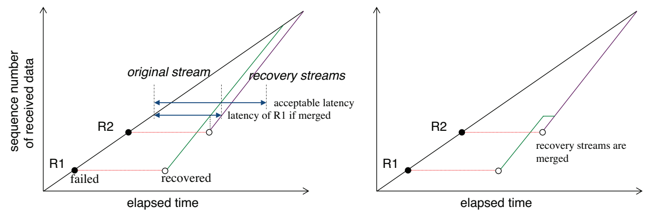
3. Preliminaries

We propose a reliable and secure multi-receiver stream delivery system that combines existing techniques, SRSM and CLAME. In this section, we describe the overview of SRSM and CLAME as preliminaries to explain our system design.





(a) failure and recovery situation (b) after SRSM merging



(a) failure and recovery situation (b) after SRSM merging

Fig. 3 SRSM case II

3.1 SRSM

In this subsection, the basic behavior of SRSM is briefly described. SRSM is a multicast scheme that treats recoveries of lost data caused by failures.

Fig. 1 shows the system model of SRSM. The principle components of the data stream sender consist of sensor data obtainer, sensor database, and sensor data stream generator. We assume that such a sender components typically run on the gateways of the sensor network, which have larger computation and storage capabilities.

The sensor data obtainer obtains the sensed data from sensors periodically and continuously. All obtained data are stored on 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.

When a receiver encountered a failure and then recovered, 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 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 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, if the application needs to analyze slow movements of objects in the video frames, some frames can be skipped. Thus the recovery stream can be delivered faster than the original stream. Once the recovery stream catches up with the original stream, then the recovery stream is merged.

In the following, we denote R1 and R2 as two receivers encountering different loss conditions. 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 may depend on the purpose of receivers. For example, the receivers who use the data for archiving applications can tolerate long latency. However, longer latency is not acceptable for the receivers of real-time applications such as controlling robots.

If there is one or more recovery streams, SRSM merges the recovery streams to alleviate bandwidth usage on the sender. There are two types of merge situations. The two cases are shown in **Fig. 2** and **Fig. 3**. In these figures, x-axis shows the elapsed time and y-axis shows the sequence number of received data. The black line shows the original stream and colored lines show the recovery streams.

In Fig. 2, there are two recovery streams. The earlier recovery stream for receiver (R1) possesses lower sequence number than that of the receiver (R2), at the time R2 was recovered. Therefore, SRSM checks the acceptable latency of receiver (R2). Taking into account the time when the recovery stream for receiver (R1) delivers the data sequence number requested by receiver (R2), SRSM calculates the actual latency of the recovery stream with the larger sequence number of received data. If the actual latency satisfies the acceptable latency of receiver (R2), SRSM merges the two recovery streams. As a result, the receiver (R2) is merged. Likewise, in the Fig. 3, R1 is merged into the stream received by R2 considering the acceptable latency. If the acceptance latency is not satisfied, the recovery streams are not merged.

3.2 CLAME

In this subsection, the basic behavior of CLAME, which achieves requirements including authentication, confidentiality, message integrity, and replay attack prevention for multi-receiver data delivery, is briefly described. We apply CLAME for shared secret renewal process because it has more security features than other schemes and has better performance. CLAME is suitable for IoT devices that do not have plenty of CPU powers because the encryption and decryption computation load is smaller than other schemes. The basis of the CLAME is as follows:

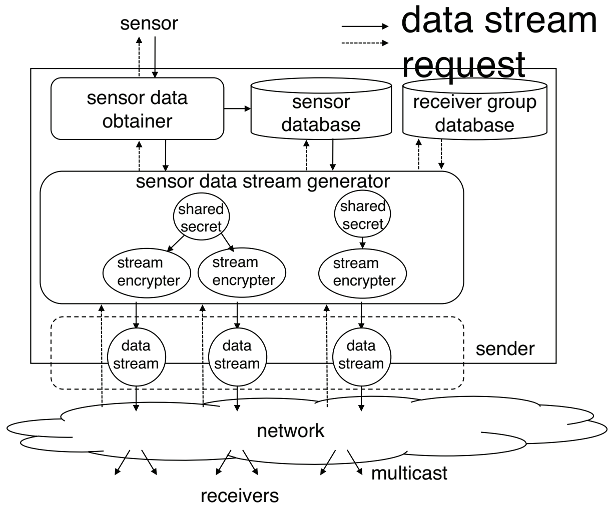


Fig. 4 Proposal system model

- For ensuring authentication, in the proposed CLAME, trusted third party provides partial private keys. Although trusted third party helps to authenticate users, it is not allowed to reveal the data.
- To prevent sensitive data leakage from trusted third party, each user needs to generate own key pair (public key and private key).
- Confidentiality is achieved by encryption. In the case of encryption, public keys of trusted third party and intended receivers are the main components to protect sensitive data from revealing by unintended receivers.

The basis of the CLAME is as follows:

- Setup: Trusted third party generates public parameters *params* and its master secret key.
- Set-Key-Pair: All users (senders and receivers) generates own secret value and public key.
- Partial-Private-Key-Extract: Trusted third party generates partial private keys for all users.
- Set-Public-Key: All users (senders and receivers) modifies the public key by adding partial private key to it.
- Encrypt: Sender encrypts the message M by using *params*, public key of trusted third party and target receivers' public keys. To maintain message authentication and replay attack prevention properties, the sender generates signature. Finally, ciphertext C is given as output.
- Decrypt: Receiver takes C, *params*, partial private key and private key as input to get back plaintext message M.

In our stream delivery system, the sender delivers encrypted data stream. The shared secret for the data stream encryption is encrypted and delivered to the receivers by CLAME. As an encryption scheme, standard encryption algorithms such as AES [13] etc. can be applied. Details of the CLAME scheme have been proposed in [5].

4. 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.

4.1 System model

Fig.4 shows the system model of our proposal. The largest difference from the system model of SRSM is that the generated data streams are encrypted. The stream generator includes stream encrypter, which encrypts the stream by shared secret. The shared secret is generated for each stream multicast. The shared secret may be used by the multiple stream encrypters.

To alleviate the computation load to renew shared secret, our proposal system splits the receivers of a data stream into multiple groups and use different shared secrets for them. By splitting the receivers into smaller groups, the computation load of encrypting shared secret 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. The system needs to manage which receiver belongs to which group. To manage the receiver groups that belongs to the stream, the receiver group database exists. The details of the group management is described in the next subsection. We assume to apply CLAME as an encryption scheme but other multi-receiver encryption schemes [10], [11], [12] can also be applied.

4.2 SRSM-R

To implement a reliable and secure multi-receiver stream delivery system by simply combining SRSM and CLAME, 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 considering recovery stream deliveries, we propose an extension of SRSM, which we call SRSM-R (Synchronized Recovery Stream Merging with Receiver number limitations). In SRSM-R, the number of receivers that can be handled by one original stream are limited. In other words, an original stream is intended for a particular group of receivers. Multiple original streams are prepared to deliver data streams for receivers on each group. 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

$$G = ceil(\frac{N}{T}).$$

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 is the same as that of its original stream.

Algorithm 1 shows the pseudo code of SRSM-R describing the functions that are run on the sender. When a receiver requests to subscribe to a stream, the sender of the stream sends the shared secret and the corresponding stream identifier encrypted by CLAME as the response. The receiver then joins the multicast with the corresponding stream identifier. The receiver is assigned

Algorithm 1: SRSM-R algorithm

```
on receiving subscribe request from receiver $n$
begin
    $g \leftarrow$ find_vacant_group();
    if ($g = nil$) then
        $g \leftarrow$ new_group($\{n\}$);
        $g.secret \leftarrow$ new_shared_secret();
        $s \leftarrow$ SRSM_new_stream();
    else
        $g \leftarrow g \cup \{n\}$;
        $s \leftarrow$ SRSM_original_stream($g$);
    // notify $n$ the shared secret and stream identifier of $s$
    unicast($n$, CLAME_encrypt($\{n\}$, $g.secret + s.id$));
on receiving unsubscribe request from receiver $n$
begin
    $g \leftarrow$ find_group_contains($n$);
    $g \leftarrow g - \{n\}$;
    $g.secret \leftarrow$ new_shared_secret();
    multicast($g$, CLAME_encrypt($g$, $g.secret$));
on receiving failure recovery request from receiver $n$
begin
    $s \leftarrow nil$;
    foreach $r \in$ SRSM_recovery_streams() do
        if $r.size < T \wedge$ SRSM_mergeable_case_I($r, n$) then
            $r.members \leftarrow r.members \cup n$;
            $s \leftarrow r$;
            $break$;
    if $s = nil$ then
        $s \leftarrow$ SRSM_generate_recovery_stream();
        foreach $r \in$ SRSM_recovery_streams() do
            if $r.size < T \wedge$ SRSM_mergeable_case_II($s, r$) then
                $s.merge(r)$;
    // notify $n$ the stream identifier of $s$
    unicast($n$, AES_encrypt($g.secret$, $s.id$));
function delivery process on time $t$
begin
    foreach $g \in groups$ do
        foreach $s \in$ SRSM_all_streams($g$) do
            multicast($s.members$, AES_encrypt($g.secret$, $s.data(t)$));
```

to a group on the sender. 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. When a receiver unsubscribed from a group, the shared secret renewal process is executed and newly generated shared secret is delivered to the rest of the members by multicast with CLAME encryption. When a recovery request is received from a receiver, 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. Then, the sender delivers the stream data by multicast with AES encryption for every time slot.

4.3 Shared secret renewal in SRSM-R

In SRSM-R, the sender needs to modify the access control policy whenever any receiving member no longer wants the data or anyone of 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

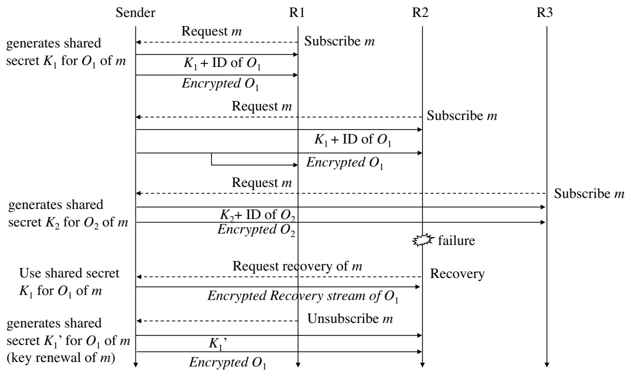


Fig. 5 Signalince sequence: Three receivers(R1, R2, R3) subscribe to m, R2 encountered failure and recovery, R1 unsubscribe from m

that only intended receivers can reveal it.

Fig. 5 shows an example sequence of the signaling to deliver 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, the threshold $T = 2$ is assumed. 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 (receivers are $R1$ and $R2$) and $K2$ for O_2 (receiver is $R3$). The sender sends two multicast data streams using corresponding shared secret. When the receiver $R2$ encounters 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 member of O_1 is changed. Then the sender generates a new shared secret K'_1, encrypts the stream data O_1 with it, and delivers the encrypted data to the remaining receivers, in this case, $R2$.

5. Implementation and Performance Evaluation

In this section, the performance of our secure multicast data delivery system is evaluated. Our evaluation is based on the measurement using an implementation of security mechanism of CLAME and simulations of SRSM-R.

5.1 An implementation of CLAME

To evaluate the performance and feasibility of the proposals, we implemented a security mechanism that follows our system design. The comparison schemes based on the bilinear pairing are implemented by using Java pairing based library [14]. We use a symmetric bilinear pairing $e : G1 \times G1 \rightarrow G2$, where $G1$ is an additive group with prime order q on super singular elliptic curve having a prime field F_p and the size of p and q are 512 bits and 160 bits respectively. To implement the same security level, for the proposed CLAME scheme, we use non-singular elliptic curve where p and q are 160 bits. And elliptic curve cryptography is implemented using bouncy castle library [15]. Evaluation is performed on nexus 7 tablet having android version 4.4.2 with

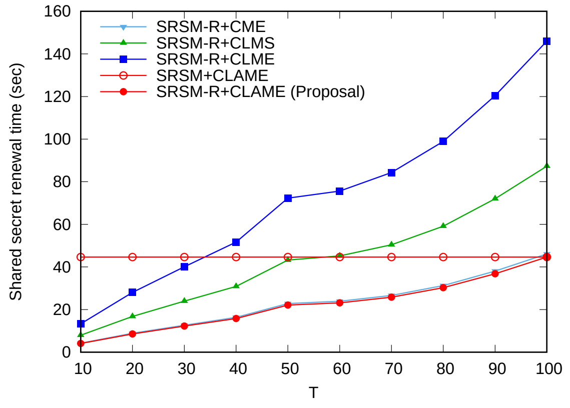


Fig. 6 Shared secret renewal time comparison: The shared secret renewal time is measured using encryption algorithms including existing schemes and CLAME.

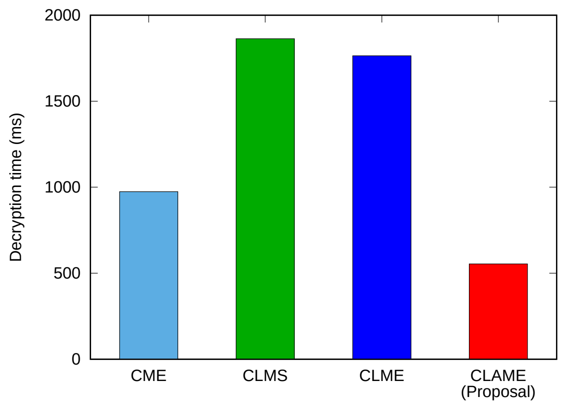


Fig. 7 Decryption time comparison: The decryption time is measured using encryption algorithms including existing schemes and CLAME.

Quad-core 1.2 GHz Cortex-A9 CPU and 1 GB RAM.

5.2 Performance Evaluation

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

Fig. 6 shows the result of shared secret renewal cost using the combination of SRSM-R and different multi-receiver encryption schemes, including existing encryption schemes and CLAME. Existing schems are denoted as CME (Certificate-based Multi-receiver Encryption) [10], CLMS (Certificate-less Multi-receiver Signcryption [11], and CLME (Certiticate-less Multi-receiver Encryption) [12]. Fig. 6 also shows a result of shared secret renewal time using combination of SRSM 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. In the experiment, the message payload size is set as 256 bits, which corresponds to the typical size of the shared secret. 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 SRSM-R requires shorter shared secret renewal time for all T values.

Fig. 7 shows a result of decryption time comparison of the encryption schemes. The y-axis shows the result of decryption time on a receiver in milliseconds. Though the shared secret renewal time of the proposed scheme and CME is almost equal, the decryption time of the proposed scheme is shorter.

The number of the streams managed by the sender is evaluated by simulations. **Fig. 8** shows the result of the average number of

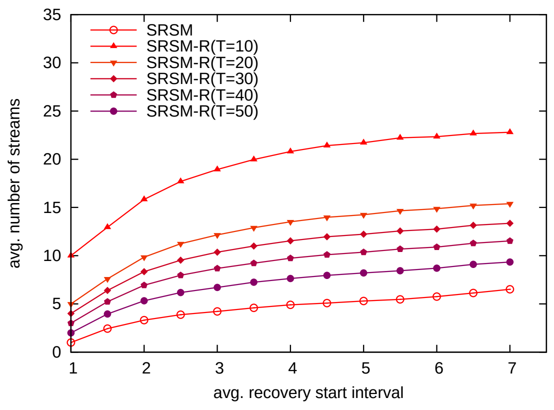


Fig. 8 Average number of streams: The loss and recovery follows the Poisson process. Average recovery start interval is changed while average loss start interval is set as 7.5.

Table 1 Simulation setups

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

streams generated for SRSM and SRSM-R, changing the threshold T from 10 to 50. The simulation setups are shown in Table 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 recovery stream progress speed is twice as the speed of the original stream within a unit time, which corresponds to the situation in which half of the stream data is skipped. 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 to see the behavior of the scheme. The failure and recovery follows the Poisson process. In the simulation, the average recovery start interval is changed while average loss start interval is set as 7.5. The larger the average recovery start interval is, the larger the loss interval becomes. In the Fig. 8, the y-axis shows the average number of streams through the simulation time and the x-axis shows the average failure recovery start interval.

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 SRSM-R. For example, if the T is set as 50, about 2.8 streams increase on average when the average recovery start interval is 5.0. Appropriate T value may differ depending on the computational ability and available network bandwidth of the sender. 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.

6. Conclusion

In this paper, 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 CLAME. To cope with the overheads of access right policy changes on CLAME, we proposed SRSM-R, 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 SRSM-R by simulations on probabilistic model. We confirmed the trade-off between the shared secret renewal cost and the number of streams.

Our future work includes further performance improvements, more detailed evaluations on the various environments, and actual implementation of the whole system.

Acknowledgments This research was supported in part by Grand-in-Aids for Scientific Research (B) numbered 15H02702, and the NICT-Osaka University joint research "Research and development of advanced network platform technology for large scale distributed computing."

References

[1] Patel, M. et al.: Mobile-Edge Computing-Introductory Technical White Paper, ETSI MEC white paper, Vol. 1, (2014).
[2] Verbelen, V., Simoens, P., Turck, F.D. and Dhoedt, B.: Cloudlets: bringing the cloud to the mobile user, *Proc. third ACM Workshop on Mobile cloud computing and services*, pp. 29–36 (2012).
[3] Teranishi, Y., Kimata, T., Yamanaka, H., Kawai, E. and Harai, H.: Dynamic Data Flow Processing in Edge Computing Environments, *Proc. Intl. Conf. on IEEE Computer Software and Applications 2017 (COMPSAC 2017)*, pp. 935–944 (2017).
[4] Ei Khaing Win, Yoshihisa, T., Yoshimasa, I., Kawakami, T., Teranishi, Y. and Shimojo, S.: A Lost Sensor Data Recovery Scheme for Faster Data Streams Merging, IPSJ SIG Technical Report, 2017-DPS-169, pp. 1–6 (2017).
[5] Ei Khaing Win, Yoshihisa, T., Yoshimasa, I., Kawakami, T., Teranishi, Y. and Shimojo, S.: A Lightweight Multi-receiver Encryption Scheme with Mutual Authentication, *Proc. IEEE International COMPSAC Workshop on Security, Trust and Privacy for Software Applications (STPSA'17)*, pp. 491–497 (2017).
[6] Hua, K.A., Cai, Y. and Sheu, S.: Patching: A multicast technique for true video-on-demand services, *Proc. sixth ACM Intl. Conf. on Multimedia*, pp. 191–200 (1998).
[7] Carter, S.W. and Long, D.D.: Improving Video-on Demand sender Efficiency Through Stream Tapping, *Proc. sixth ACM Intl. Conf. on Computer Communications and Networks*, pp. 200–207 (1997).
[8] Eager, D.L. and Vernon, M.K.: Dynamic Skyscraper Broadcasts for Video-on-Demand, *Intl. Workshop on Multimedia Information Systems*, pp. 18–32 (1998).
[9] Golubchik, L., Lui, J. and Muntz, R.:*Reducing I/O Demand in Video-on-Demand Storage senders*, ACM, Vol. 23, No. 1, pp. 25–36 (1995).
[10] Chul, S., Jung, C.D. and Rhee, K.H.: Multi-receiver Certificate-based Encryption and Application to Public Key Broadcast Encryption, *Proc. of Symposium on Bio-inspired, Learning, and Intelligent Systems for Security*, pp. 35-40 (2007).
[11] Selvi, S.S.D., Vivek, S.S., Shukla, D. and Chandrasekaran, P.R.: Efficient and Provably Secure Certificateless Multi-receiver Signcryption, *Proc. Intl. Conf. on Provable Security*, pp. 52-67 (2008).
[12] Zhu, J., Chen, L-L., Zhu, X. and Xie, L.: A New Efficient Certificate-less Multi-receiver Public Key Encryption Scheme, *Intl. J. Computer Science Issues (IJCSI)*, Vol. 13, No. 6, pp. 1–7 (2016).
[13] Selent, D.: Advanced Encryption Standard, *J. RIVIER ACADEMIC*, pp. 1–14 (2010).
[14] Caro, A.D. and Iovino, V.: jPBC: Java pairing based cryptography, *Proc. of the 16th IEEE Symposium on Computers and Communications, ISCC*, IEEE, pp. 850–855 (2011), available from ⟨http://gas.dia.unisa.it/projects/jpbc/⟩.
[15] Bouncy Castle, available from ⟨http://www.bouncycastle.org/⟩.