Received: 27 March 2022 | Revised: 4 July 2022 | Accepted: 14 July 2022 | Published on: 5 August 2022

DOI: 10.1002/spy2.258

RESEARCH ARTICLE

WILEY

Entropy service for secure real-time mission critical communications

Engin Zeydan[1] | **Yekta Turk**[2] | **Yaman Yagiz Tasbag**[3]

[1]Services as Networks (SAS) Research Unit, Centre Technologic de Telecomunicacions de Catalunya, Barcelona, Spain

[2]Mobile Network Architect, Istanbul, Turkey

[3]Computer Engineering Department, Bilkent University, Ankara, Turkey

Correspondence
Engin Zeydan, Services as Networks (SAS) Research Unit, Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), 08860 Castelldefels, Spain.
Email: engin.zeydan@cttc.cat

Funding information
European Regional Development Fund; National Program on Equipment and Scientific and Technical Infrastructure, Grant/Award Number: EQC2018-005257-P

Abstract

Real Time Mission Critical Communication (RTMCC) in emergency situations can include real-time video and audio calls between peers and first responders all occurring simultaneously. RTMCC also requires secure end-to-end (E2E) group communication (GC) sessions against potential security threats during such incidents. In this paper, we explore all aspects of the possible methods that are suitable for a software implementation of for session key change during GC in E2E encryption of RTMCC. Later, we introduce our *Entropy Service* concept, which can be very effective in secure E2E RTMCC sessions. The proposed method ensures E2E security in real-time communication systems while allowing very fast session key change for clients involved in an RTMCC session with a computational complexity of $\mathcal{O}(1)$. Our experimental results show that the proposed *Entropy Service* can reduce total time by 99.6% and 99.2%, the idle time by 99.4% and 98.99%, and the number of messages by 51.4% and 35.33% compared to the key refreshing and hash methods, respectively, when the number of users in the system increases to 45. These results show that both communication and computation complexity are significantly reduced with the proposed RTMCC session key change.

KEYWORDS

entropy, mission critical, real time, security, service

1 | INTRODUCTION

A mission critical (MC) application can help prevent successful completion of an intended operation that may be potentially dangerous. There are many applications of MC systems in rescue, military, search and recovery (mountain/sea), forest firefighting, disaster and emergency management. Any disruption to MC applications can have direct financial or life-threatening consequences. At the same time, real-time mission critical communication (RTMCC) technologies that connect clients in different locations have recently gained extreme importance (eg, in autonomous cars and avionics). In RTMCC applications, many clients can participate in single group communication (GC) session. This provides great flexibility for industries such as public safety agencies, critical infrastructure (utilities, railroads, and power grid operators) and governments providing mission-critical services. In the market today, there are well-known GC applications such as Skype, Webex, and so on, that have been widely deployed and in use for a long time. Unfortunately, although these applications provide a communication platform with their audio/video calling capabilities, they cannot be considered a comprehensive solution for GC, especially in the context of a MC service that relies on low-latency communication.

Various RTMCC services may be present in the portfolio of service providers (SPs) and mobile network operators (MNOs), which are expected to be diverse in 5G and 6G networks. For example, augmented reality (AR)-assisted virtual dispatchers can be used by first responders to obtain comprehensive information for on-site decision making, autonomous machines can perform tasks without human intervention, objects can be identified and analyzed in real time using cameras and sensors, for example, for police special operations, and so on. In RTMCC-based applications, from the end-user's perspective, a GC session is first created by an administrator. Then, other users are invited to participate in the GC session. The invited users can accept or decline to participate in the proposed GC session. During the GC session, both the user's video and the video received from the remote clients are displayed together in the application.

At the same time, RTMCC-based applications should also be secure, resilient and trustworthy, and their performance should be enhanced with critical security features to protect against serious errors in MC applications. Therefore, data encryption, client authentication, and per-IP packet integrity checking are crucial in GC sessions.[1] One important method of restricting access to information is encryption and the selective distribution of keys to encrypt group information. However, this requirement also presents some challenges, as it can be difficult to achieve a high level of security for a large number of controlled users with simple solutions. Secure RTMCC sessions can be created if an encryption technique is applied that takes the input group message in RTMCC sessions and uses the *cryptographic key* to perform some transformations to produce cipher text. In this way, the messages can be secured with the selected group key, which is known only to the clients participating in the RTMCC session. However, the clients participating in the RTMCC session may change dynamically over time, making it necessary to update the selected group key. Otherwise, the leaving or excluded group member can continue to access the group communication. To prevent this, a new key is distributed when a new client joins the GC session (or an old client leaves the RTMCC session) so that the new (old) client cannot decrypt the previous messages.

One of the security challenges in providing an effective RTMCC session is to control GC access to RTMCC sessions and their information as efficiently, easily and scalable as possible. In RTMCC session, distributing the group key to valid clients can be a complex problem.[2] As the session membership changes very frequently, an RTMCC session becomes difficult to administrate. For example, there can be problems with rekeying the group, especially when a client leaves the RTMCC session. This is because the old key cannot be used within the RTMCC because the leaving client already knows the old key. For this reason, the master key distributor of the RTMCC session should be able to provide secure keys to the group in a simple, scalable, and secure manner. Although simple solutions can be used, the cost of using such solutions in large groups can also be very high.

Asymmetric cryptography seems to be slow, especially when it comes to real-time communication.[3] For this reason, using symmetric cryptography methods for encryption during RTMCC sessions is considered more powerful.[4] To encrypt/decrypt the GC session, all clients must have the same symmetric encryption key that is used during the session. One simple scheme to rekey the group is to assign a secret key to each client of the group via key distribution center (KDC). When the distribution of this session key is handled by a server, it is basically called a server-based security mechanism. However, this operation is not scalable and simple. It can also be costly since KDC has to encrypt the group key n times. Additionally only after clients mutually agree on this session key and the key is not kept on the server, end-to-end (E2E) encryption can be achieved. In this paper, we aim to reduce the delay and the number of messages exchanged during session symmetric session key (SSK) changes in a given GC session for RTMCC applications, where more RTMCC users can be served simultaneously with minimal overhead while ensuring E2E encryption.

1.1 | Related work

1.1.1 | Studies on group-key management

First responders usually work in groups and need group communication to organize their tasks efficiently. In general, end-to-end media security is much more complex for group calls than one-to-one calls. In the literature, there are many studies on Group Key Management (GKM). The surveys in References 2,5,6 examine the various aspects of GKM. In Reference 2, the authors define three groups, mainly centralized group key management protocols, decentralized architectures and distributed key management protocols. The concept of distributed key management is based on all clients agreeing on one SSK in the group. In decentralized key management, nodes are selected as the KDC and in centralized key management, a central server is positioned as KDC. The study in Reference 7 explains one-way function trees and chains for providing keys to large domains. None of these studies have an E2E security view as all of these three defined groups are based on a KDC positioned in a group.

There are also already solutions for managing post-compromise security and asynchronous group key establishment with forward secrecy without (n) or $\mathcal{O}(n^2)$ operations for messaging applications. For example, the key trees used in the messaging layer security (MLS) protocol are an example of this Reference 8. However, these protocols are designed for asynchronous communications and not for synchronous communications. Moreover, these protocols in future messaging applications are not yet foreseeable in practical scenarios.

These studies on GKM can be considered similar to the method we propose in this paper, but there are concrete structural differences arising from the nature of the RTMCC use case. First, RTMCC has fundamentally different architectural components such as the group management signaling server and the traversal using relays around NAT (TURN) server for communication. Our proposal is based on the use of the signaling server present in the RTMCC structure and the entropy values already attached to the messages sent by the server. The definition of an entropy service is another fundamental difference. On the other hand, the group concept defined in GKM refers to the clients in a multicast-based data transmission and has a peer-to-multi-peer messaging capability. The requirement for clients to connect to a TURN server in RTMCC overrides the tree construction used in the GKM studies. Mandatory communication through a TURN server makes parallelization of SSK distribution inapplicable with respect to real-world implementations and synchronization of RTMCC clients inapplicable.

The experimental standard documents of Reference 9 describe the architecture and specifications for symmetric cryptographic keys management for multicast communications. The structure in multicast groups may indeed have a parallelism with the clients participating in a GC. The reason for this similarity is again that a key is created and distributed to the group. However, in this structure there is peer review and a cooperative key generation structure for other parties to create a key, which is quite different from our entropy service solution. In Reference 10, application-specific key management techniques are described, but in this paper, a KDC is responsible for key distribution. In this study, we focus on maintaining E2E security, and the SSK information is known only to the clients.

1.1.2 | E2E Security efforts for real time communications and mission critical services

There are numerous papers in the literature on real-time communications, mission-critical services, and their security features in the literature.[11-13] The third generation partnership project (3GPP) has defined the structure of the private keys, their transport method, and the distribution architecture for RTMCC in mobile networks.[14] In this 3GPP study, the use of multimedia internet key exchange (MIKEY)-SAKKE key encapsulation for secure session key delivery is presented. Unfortunately, the 3GPP study does not specify or recommend a key changing method for RTMCC.

The study in Reference 15 presents a self-contained public key management scheme, a scalable method of cryptographic key management that incurs almost no communication overhead for authentication. The proposed method provides high service availability, but is also suitable for wireless ad-hoc networks. The survey in Reference 16 evaluated key management schemes for the smart grid. The integration of Network Functions Virtualization (NFV) and Software Defined Networking (SDN)-based techniques can be used to solve E2E security problems.[12,13] On the other hand, congestion control is particularly important to ensure that the network functions properly while providing E2E security.[17] The following summarizes some of the relevant research, industry and standardization efforts focused on four distinct categories:

i Commercial products: Although there are many Web Real-Time Communication (WebRTC)-based applications on the market, few of them have the E2E encryption feature. For Google Duo, one of the applications that use E2E encryption, the technical document in Reference 18 describes the method used for the SSK changing. Google Duo uses the *hash functions method* for the joining user and the *key refreshing method* for the exiting user and runs purely event-driven. The recently issued US patent in Reference 19 describes the key refreshing method and is held by WhatsApp. This patent specifies the key refreshing procedure for RTMCC systems and also defines the client selection process responsible for distributing the SSK. In this case, it would be reasonable to assume that WhatsApp works entirely with the key refreshing method.

ii Open source applications: There are also open-source Real Time Communication (RTC) applications, Jitsi and Signal,[20,21] which have recently started supporting E2E encryption. The SSK changing structure in Jitsi and Signal is the same as that used by Google Duo's SSK changing method. Janus[22] has a test implementation for E2E encryption and its technology is not yet mature.

iii Standardization efforts: MC service requirements from the public safety (PS) domain are expressed in the latest 3GPP releases. Many enablers for mission critical networks are already standardized for 5G in 3GPP and some will be finalized in Release 17. For example, 3GPP Release 13 introduces the mission-critical push-to-talk (MCPTT) service for voice communication,[23] Release 14 introduces the mission critical video (MCVideo)[24] and mission critical data (MCData)[25] services that enable video communication and data streaming functionalities to public safety users respectively. The mission critical security architecture, that protects metadata and communications of mission critical services and provides both signaling and application plane security mechanisms, is currently being studied by 3GPP TS 33.180.[26] Standardization efforts to integrate mission-critical push-to-X services into the network are planned for Release 18.

iv Research projects: The European Union (EU) H2020 5G ESSENCE project* addresses the management of network services to provide MC public safety services. Another EU funded project, DARLENE,† aims to deploy technologies (eg, AR glasses) for law enforcement agencies (LEAs) and first responders to make more informed and faster decisions, especially in mission-critical situations where time is of the essence. Another EU-funded project RESPOND-A (Next-generation equipment tools and mission-critical strategies for First Responders)‡ is working to develop technologies based on 5G, AR, Virtual Reality (VR) and autonomous robots to optimize the prediction and assessment of incidents by first responders during and after disasters. Another EU-funded project CyberSANE,§ aims to improve security and resilience of critical infrastructures in health, energy, and transport and to enable more dynamic and collaborative warning and response systems.

1.2 Our contributions

In Reference 27, the defined cryptographic message format can be used to secure the symmetric key. There is a word-of-the-day field in this message format, but this field is not used for the entropy source and is used to preserve the freshness of the cryptographic messages. The recently published National Institute of Standards and Technology (NIST) document in Reference 28 mentions cryptographic services such as the random number generation service, but unfortunately these statements focus on local services and again do not address E2E security use cases. The *Entropy as a Service* concept was also introduced by NIST in 2016[29] and can be evaluated as not yet mature. Although this concept has a high-level architecture, it is about decentralizing the root of trust in a network. Moreover, there are no defined studies on its use cases, especially in the RTMCC domain.

Validation of an RTMCC service requires evaluation of the signaling logic, media connectivity, quality-of-service (QoS), characteristics of large, complex, distributed, and heterogeneous systems.[30] In our study, we apply our proposed entropy service to the RTMCC domain, and the impact of our proposed method directly affects the video and audio quality of a GC. The long completion time for session key changing in the GC may result in packet and session loss. We show how entropy messages can be transmitted efficiently and cost-effectively using the group management signaling server, which can improve response times and increase the daily efficiency of RTMCC services. We also provide additional implementation capabilities. Our main contributions to this study can be described as follows:

- We present key changing procedures for E2E security in RTMCC and the main reasons for requesting key changing in encrypted RTMCC sessions.
- We examine the current and possible procedures for changing RTMCC session keys, and also show under what circumstances the other procedures perform key changes. More specifically, we explain the existing problems of these procedures.
- We describe our *Entropy Service* solution and explain its procedure in detail. The performance of the proposed method in experimental tests shows that the proposed *Entropy Service* can reduce 99.6% and 99.2% of the total time, 99.4% and 98.99% of idle time, and 51.4% and 35.33% of the number of messages compared to the *Key Refreshing* and *Hash functions methods* respectively, when the number of users in the system increases to 45.
- With the proposed Entropy Service method, there is no packet loss during the session key change while the highest quality level of GC can be achieved.

The remainder of the paper is organized as follows: In Section 2, we explain the background of current RTMCC technologies and provide an overview of the end-to-end security architecture of RTMCC. Section 3 introduces the event-based methods used in our evaluations. In Section 4, we present our entropy service solution with design guidelines.

TABLE 1 Symbols used throughout the paper

Symbol	Meaning
Entropy	Entropy value
freq_join	frequency of client join to GC
threshold	threshold set for frequency of client joins to GC
Message	Data that is transmitted/received for each client
n	triggering interval
SSK_i	SSK value at i-th counter
Synch	Server synchronization check Boolean value
Hash(x)	Hash function for input x
Gen()	Generator function
Dist()	Function for message distribution to clients

Experimental analysis in Section 5 shows the advantages of our proposed method. Finally, in Section 6, we present the conclusions of the paper. Table 1 provides a list of symbols used in this paper.

2 | USE CASE, ARCHITECTURE AND BACKGROUND

2.1 | RTMCC use case

In our scenario, we consider use cases in which emergency situations such as natural disasters, utility failure, health crises, vehicular accidents have occurred. During an emergency situation, a commander leading rescue operations must communicate (video and audio) securely and in real time with the response team and civilians, and may also need assistance in receiving video images from drones and other cameras to display on a map or AR screen. For this reason, when providing MC services, real-time group calling, private calling, and emergency calling features should be enabled, and sessions should be E2E encrypted for security and privacy reasons. These reliable communications and the network used by the MC service can be critical, as the difference between a reliable and an unreliable network can be a matter of life and death in some cases. One of the other reasons for using security mechanisms for communication channels is that communications between public safety organizations, personnel, or appropriate command and control centers contain sensitive information, such as victim names, injuries, and other relevant information that should only be accessible to authorized personnel. For this reason, the confidentiality of data transmission via on-site devices should always be ensured during this cross-communication.

MC services require that security control teams quickly gain situational awareness (eg, of unattended or dangerous items, the number of people in a crowded space, and illegal activity by people at surveillance sites) and make quick decisions to address specific tasks. In this regard, secure low-latency RTMCC services provided in crowded areas (eg, airports, stadiums, and shopping malls) between teams of mobile security officers and security operations centre (SOC) operators can help improve situational awareness (ie, understand the environment and facilitate coordinated response and tactical action planning by law enforcement in the event of an active dangerous incident). Therefore, in the event of an incident or extreme situation, the RTMCC can help expedite mission-critical operations, improve situational awareness for crisis management, and improve health care provision (assisting first responders and health care personnel in providing care and assistance) to protect citizens from injury and potential danger. Therefore, secure, reliable and real-time communications can help clearly identify crisis events so that assistance can be provided rapidly and decisions can be made more quickly.

2.2 | RTMCC security architecture

Figure 1 shows the high-level design of the RTMCC security architecture in general form for a given RTMCC system with three categories where each of them has first responders such as police, firefighters, and ambulance, and rescue services.

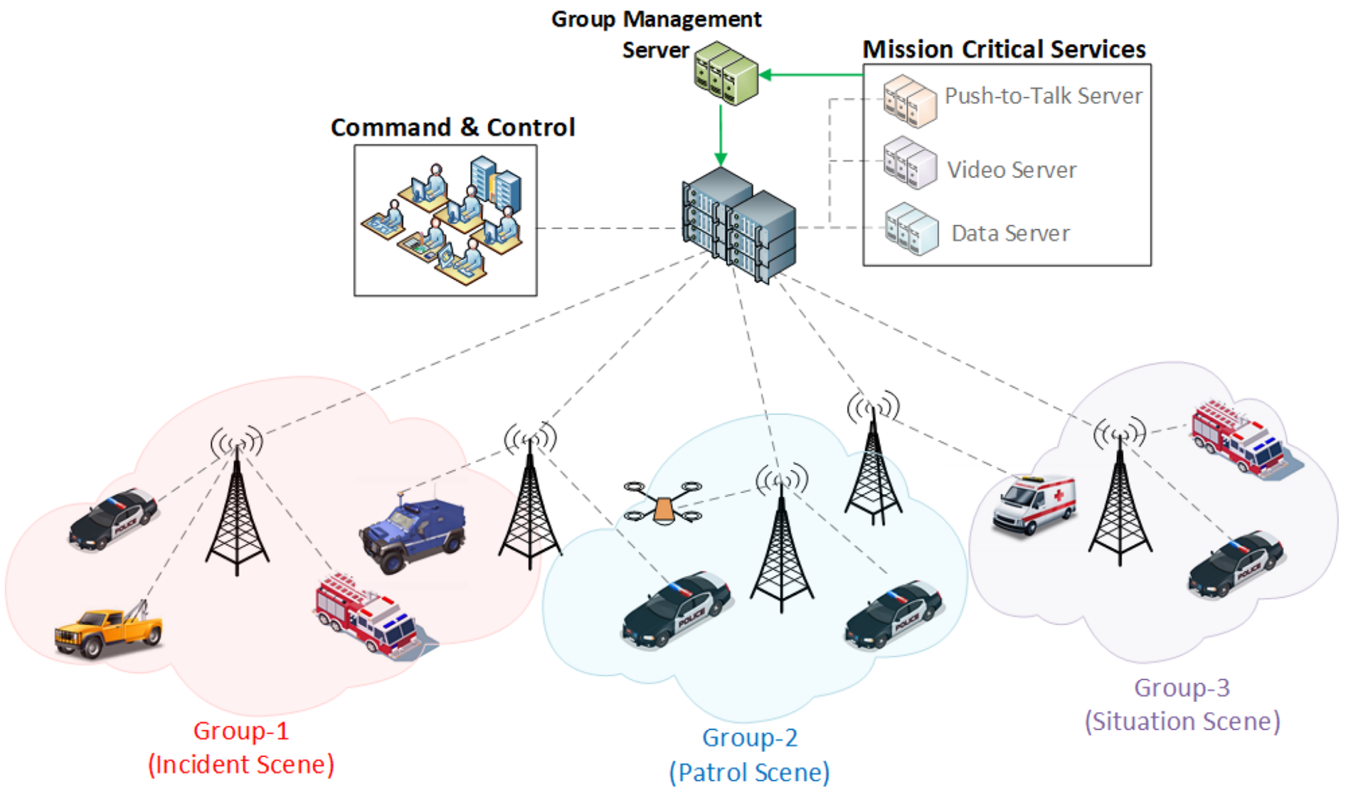

FIGURE 1 High-level view of mission critical secure services provided to users belonging to different groups

For example, RTMCC system in Group-1 is for an incident (eg, emergency situations involving police officers, firefighting squads and emergency medical services requiring transition to more secure and reliable communications due to local events such as fires or, accidents), Group-2 is for a patrol scene (eg, clients using drones and other vehicles for enhanced visibility and damage assessment and inspection for surveillance), and Group-3 is for a situation scene (eg, public safety clients that need to be equipped with prioritized RTMCC systems and network connectivity, eg, due to large regional events such as earthquakes or floods). The RTMCC system security architecture provides protection between MC groups and clients. Figure 1 shows a group management signaling server, mission-critical services (push-to-talk, video, and data services), and the RTMCC clients involved in the communication. The group management signaling server confirms the suitability of the call. Once approved, a media plane can be established between them in a given GC session.

2.3 | Security overview of RTMCC

Figure 2 shows the RTMCC security structure with transport layer security (TLS) for the signaling channel and datagram transport layer security (DTLS) for the media plane. DTLS is a stream-oriented protocol that is supported by most web browsers in a built-in structure. This built-in structure of DTLS is ideal for encrypting the RTMCC data plane.[31] Moreover, no prior setup is required before a DTLS is established between peers. Therefore, data plane communication (between clients) via DTLS sessions is done in a peer-to-peer (P2P) and secure manner. Another requirement is to establish secure connections between clients and the signaling server. For this reason, a TLS session is established to secure the signaling messages. Therefore, the signaling messages (session initiation protocol (SIP), eXtensible Messaging and Presence Protocol (XMPP), etc.) are now securely transmitted within this TLS session. Consequently, the signaling channels and data planes are secured via a TLS and DTLS connection, respectively.

The focus of the RTMCC protocol is entirely on the specification of the media plane, while the signaling channel is left as much as possible to the RTMCC application developer. Therefore, RTMCC applications can use various standardized signaling protocols such as SIP, XMPP, and so on. This means that there is no defined standardized signaling channel structure in RTMCC.[32] The media plane is provided over the Internet and clients send media fragments over

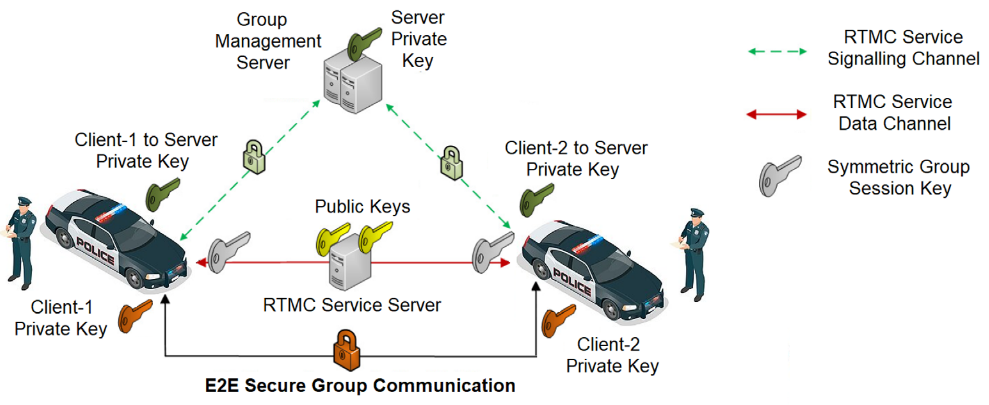
Group Management Server
Server Private Key
RTMC Service Signalling Channel
RTMC Service Data Channel
Symmetric Group Session Key
Client-1 to Server Private Key
Client-2 to Server Private Key
Public Keys
RTMC Service Server
Client-1 Private Key
Client-2 Private Key
E2E Secure Group Communication

FIGURE 2 Security structure of the RTMCC with TLS for signaling channel and DTLS for the media plane

the media gateway using real-time transport protocol (RTP). In situations where clients have different internal network internet protocol (IP) addresses because they are behind firewalls or network address translations (NATs), certain network policies cause connectivity issues for RTMCC communication. To address this issue, the interactive connectivity establishment (ICE) protocol is used to determine the possible communication options between a client and the media gateway of RTMCC.

Private and public keys for DTLS and TLS sessions used in the public key infrastructure (PKI) infrastructure are also shown in Figure 2. The signaling server establishes a separate TLS session with Client-1 and Client-2. To maintain a TLS session for the signaling channel, a key exchange method such as internet key exchange (IKE) can be used between the clients and the signaling server. For the DTLS session, the public keys of Client-1 and Client-2 are stored on the TURN server. An important point here is that secure real-time protocol (SRTP) is used over the DTLS session to encrypt traffic at the media plane. Key generation can be done via one of the selected clients or alternatively by the GC administrator. Instead, the DTLS session is used to distribute the generated session key to other clients that will participate in the GC. Then, the SRTP used for the media plane transmission performs the encryption over this SSK through all clients.

In summary, for media plane security, a DTLS connection is established and the SSK is securely shared over this DTLS connection. Thus, the media plane is encrypted with the SSK. An important detail is that the media plane runs over the DTLS session. SRTP encrypts the media plane over the established SSK using DTLS. Note that the key changing methods mentioned in this study refer to this SSK used for encryption/decryption at the media plane. E2E security methods in asynchronous messaging systems are outside the scope of this study.

2.4 | Changing symmetric key

There is a need to change the SSK as far as the media plane is concerned. The reason for this is the possibility that a client that has the key can save the conversation using the man-in-the-middle method after it has left the conversation. This client recording the conversation can listen to the GC session after it has left the conversation because it already has the session key. To prevent this, the SSK must be changed every time a client leaves the session. The same applies when a client is involved in a GC conversation. The symmetric session key must be changed again so that a new client added to the session does not have access to the previous conversations.

2.5 | Key ratcheting

Key ratcheting is a cryptographic method that can only move forward (and not backward) and derives a new key for each message, one step at a time. Key ratcheting is commonly used for messaging applications.[33] The Double Ratchet algorithm[34] is the extension of the protocol and was developed to enable E2E encryption for instant messaging applications. Key ratcheting is used for the instant messaging part of Signal, WhatsApp and also in Facebook Messenger, Google Allo, etc. The applications that use Key ratcheting are based on asynchronous communication. However, in RTMCC, media and voice communication are synchronous communication types, so Key ratcheting is not considered as a good solution for RTMCC.

3 | EXISTING EVENT DRIVEN KEY CHANGING METHODS

There are three important event-driven methods, *Key Refreshing*, *Hash Functions*, and *Key Hierarchy*. In the event-driven methods, the event is defined as client joining and leaving the GC. The difference between these two methods occurs when a client joins the GC. The hash functions method cannot be used when a client leaves the GC. The reason for this is explained in the following subsections. When the client leaves the GC, a new key refreshing must necessarily be performed.

3.1 | Key refreshing

The first viable method is to create a new key and distribute it to all users, that is, key refreshing. However, since the system is designed for E2E security, a client must generate the key and distribute it to others via the previously established DTLS session. The signaling server detects whether the defined join/leave events of the GC have occurred. When a client joins a session, it must send a message to the signaling server that it is joining the session. The signaling server in turn sends this message to all other clients to inform them that a new client has joined the session.

Figure 3A shows the key refreshing scheme when Client-5 joins the session. In Figure 3A, the new SSK is generated by Client-1 and distributed to the other clients by sending direct messages to each of them. Figure 3B shows the same scheme when Client-5 leaves the session. In this case, Client-1 distributes the newly generated SSK to the remaining clients in the GC. In addition, the number of messages with the new SSK sent from Client-1 to the clients can be reduced if a hierarchical tree structure is used. For example, Client-1 sends the new SSK to Client-2 and Client-3, then Client-2 and Client-3 send the SSK to Client-4 and Client-5, respectively. The use of this type of hierarchical tree structure depends on the implementation of the RTMCC software. Although this type of structure reduces the load on Client-1 when sending messages, it does not reduce the total number of messages.

In general, the SSK is created and distributed by the client that initiates the conference. If that client leaves the GC at its own request or due to network, software, etc. problems, another selected client is responsible for key generation and distribution. The signaling server can select this backup client to be responsible for the SSK. This selection must be made beforehand, that is, once the client that initiated the GC leaves, the new client should take over the task in case of join/leave events. So, the GC session should not be affected.

3.2 | Hash functions method

New SSK can be created using hash functions method. The new SSK is created by passing the previous SSK through a hash function. In this case, all clients in GC must know this hash function. This knowledge can be provided by implementing the RTMCC software. Thus, when a new client joins GC, the signaling server notifies each client of the arrival of the new client. Then, each client creates the new SSK by passing the previous key through the specified hash function. The new client joining the GC can get the hash function information, but it cannot create the new SSK because it does not know the previous SSK. The newly created SSK is forwarded to this new client by a client responsible for key distribution. Unlike the key refreshing method, all clients can now create the new SSK. In this case, a client (usually the one that initiated the GC) must forward the newly created SSK only to the new client that joins the GC.

This process is illustrated in Figure 4. As shown in Figure 4, a new client (Client-5) joins the GC. The signaling server sends this information to all clients. Then, all clients create the new SSK after passing it through the hash function. At the same time, Client-1 transmits the new SSK to Client-5. Unfortunately, when a client leaves GC, using the hash function is useless. This is because the client leaving GC is able to create the new SSK, since the client already knows the hash function used and the previous SSK.

Enabling a new hash function: Once a client leaves the session, a new hash function can be enabled for the remaining clients. This way, the client that left the GC session no longer has the hash function information. However, to synchronize this situation, multiple hash functions must be implemented and maintained in the RTMCC software on the clients. This

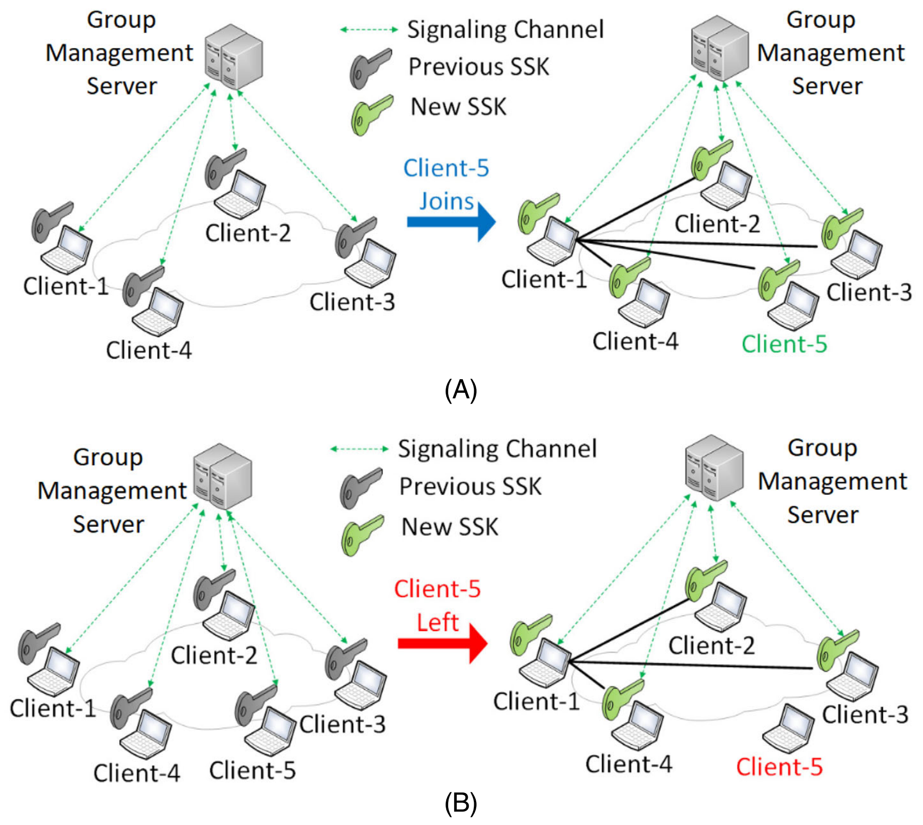

FIGURE 3 The processes of the key refreshing scheme (A) Client-5 joins to the GC session (B) Client-5 leaves the GC session

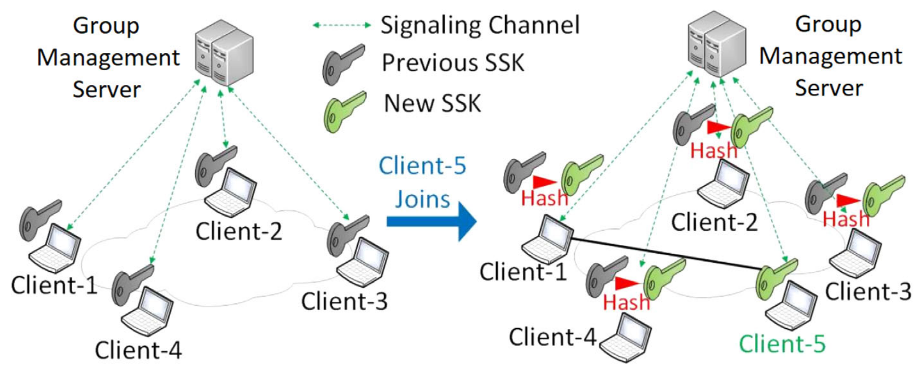

FIGURE 4 The processes of the using pre-determined hash function in each client attended to the GC and SSK delivery to the new Client-5

imposes additional costs and burdens on the RTMCC software. Maintaining dozens of hash functions in the software itself comes at a very high cost to the RTMCC software resources. Moreover, the number of possibilities is limited by the occupied memory of the alternative hash functions (Secure Hash Algorithm (SHA)-256, SHA-384, SHA-512, etc.). Therefore, the client that left the GC session can try all these possibilities to get the new SSK.

3.3 | Key hierarchy

In this approach, a KDC manages a tree of keys. The encryption keys are stored in the nodes of the tree. Each leaf of the tree represents a member of the group and contains a key encapsulation key (KED) specific to that member. Each member

TABLE 2 Complexity comparisons of event-driven methods

	Client joins		Client leaves	
Method	**Communication complexity**	**Computation complexity**	**Communication complexity**	**Computation complexity**
Key refreshing	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Hash functions	$\mathcal{O}(1)$	$\mathcal{O}(1)$	Inapplicable. Key refreshing is applied. Same complexity as key refreshing	
Key hierarchy	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$
Proposed entropy service	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

receives and maintains a copy of the KED associated with its leaf, as well as the KEDs corresponding to each node on the path from its parent leaf to the root. The key held by the root of the tree is the group key.

3.4 | Complexity of the methods

In analyzing the complexity of the two methods, we examine two aspects. The first aspect is computational complexity. The computational complexity expresses the total computation that all clients participating in GC must perform. The second aspect is the communication complexity, which is defined as the total number of message transmissions between all clients in GC. In the key hierarchy method, for a balanced tree, each member stores at most $(log2n) + 1$ keys, where $(log2n)$ is the height of the tree. Therefore, the complexity for joining and leaving GC is the same, namely $(logn)$. For the other methods, two complexity analyses are evaluated separately for the cases where clients join/leave GC.

Joining case of a new client: To evaluate the complexity of the key refreshing method, we specify a client that is responsible for generating and distributing SSK, and this client is called *Client-R*. When a new client joins the GC, *Client-R* sends the message with the new SSK to all n clients. Since this is done with at least n messages, the communication complexity value for *Client-R* can be computed as (n). To transmit these messages, *Client-R* simultaneously encrypts each message separately with the public key of the other clients. In this case, the computational complexity due to encryption with these public keys can be given as (n).

When calculating the complexity with the *Hash Functions* method, not only *Client-R*, but also all other users create the new key by processing with the hash function. *Client-R* will forward only one message to the newly added user. In this case, the number of messages to be sent is actually one and the communication complexity can be calculated as (1). Since encryption is performed using only the public key of the newly joined client, *Client-R* needs to perform only one computation and the computational complexity value can be specified as $\mathcal{O}(1)$.

Leaving case of a client: When a client leaves the GC in the key refreshing method, the complexity computation is the same as in the case when the user joins the GC. However, the hash functions method cannot be used when a client leaves the GC. When a client leaves the GC, a forced refresh takes place, and the complexity calculation is the same as in key refreshing method.

In summary, Table 2 shows the comparison of the complexity of the event-driven methods.

4 | PROPOSED ENTROPY SERVICE

In this section, we present our entropy service solution for fast SSK changing. The *hash functions* method has a value of (1) as the computation and communication complexity in the client connection and this method can be evaluated as an advanced method for the client connection case. However, unfortunately, the *hash functions* method cannot be used when the client is leaving the GC. Although the proposed *Entropy Service* method can be used for joining clients, the service shows its most important effect for clients leaving the GC. However, to explain our solution, we need to provide some information about the functions and role of the RTMCC group management signaling server. The reason is that the proposed *Entropy Service* is run by default through the group management signaling server. Moreover, clients do not have the option to register with or leave this entropy service.

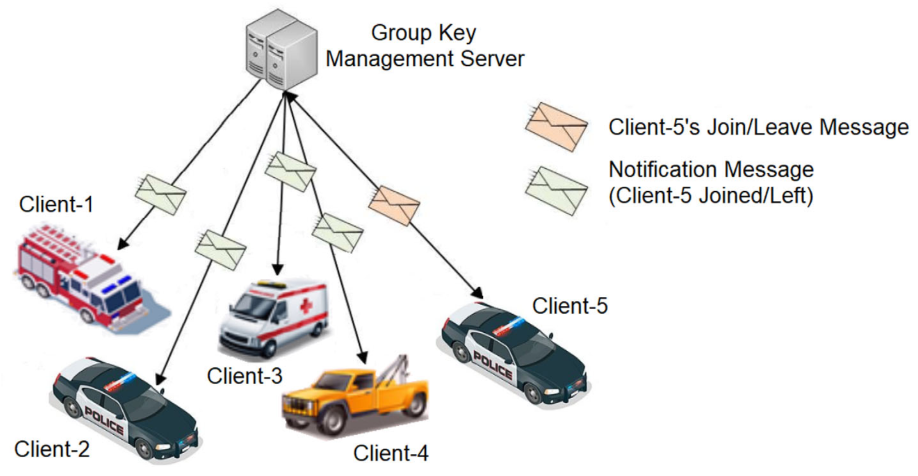

FIGURE 5 The group key management signaling server gets the join/leave messages from a client and notifies all other clients in the GC

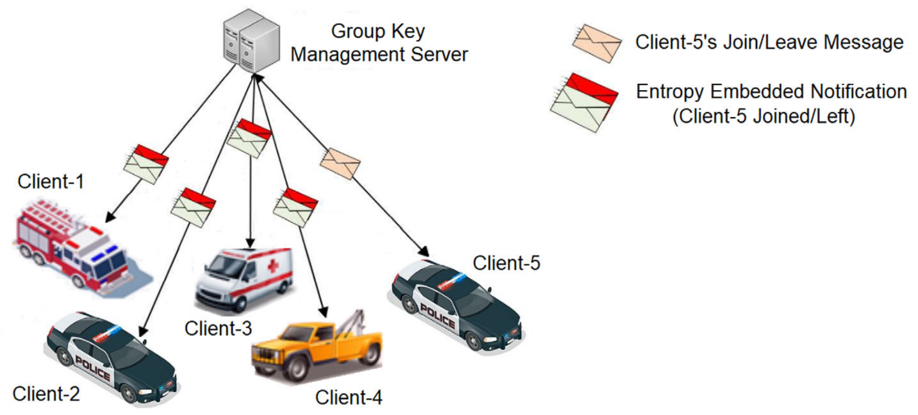

FIGURE 6 The entropy service embeds the entropy values to the notification messages

4.1 | The role of the signaling server

We mentioned in Section 2.2 that RTMCC has a signaling server in its architecture and that it is the task of the signaling server to reserve resources before the GC call is established. Event-driven methods operate depending on the joining/leaving activities of the client. These methods are notified about these join/leave events via the signaling server. Namely, when a client joins or leaves the GC, it sends an *init* or *finish* style message to the signaling server. The signaling server receives this message and then forwards it to all other clients in the GC. This initiates the SSK change by ensuring that the event-driven methods are working. This notification from the signaling server is a message that must occur in all protocols regardless of the signaling protocol (SIP, XMPP, etc.). This process of the group key management signaling server is shown in Figure 5.

4.2 | Using notification messages

After the signaling server receives a message from the joining/leaving client, it adds an entropy value that it generates and embeds in an appropriate field within those messages. An important point here is that the communication between the signaling server and the clients is secured with TLS, as described in Section 2.3 These TLS connections are established individually for each client. Figure 6 shows the use of a group key management signaling server to embed entropy values in the notification messages.

This entropy value is in fact equivalent to a random number. To create a large space for random number generation, it is ideal to generate a random number with a length of at least 256 bits. This way, no additional messaging is required to send entropy values beyond those already sent by the signaling server. Thus, the entropy value is transmitted to the clients without any additional communication or computation overhead. Since clients joining or leaving the GC do not receive notification messages, they cannot create a new SSK.

For the XMPP-based signaling implementations, the entropy value can be defined as an additional attribute field to the currently defined Extensible Markup Language (XML) schema. In our proposal, we also take advantage of the fact that the RTMCC signaling channel is not yet standardized.[32]

4.3 | Security enhancements for group management signaling server

Within the central processing unit (CPU) of the signaling server, memory encryption enables secure computation of user-level applications in the operating system (OS) code with predefined memory regions. These parts of memory are called enclaves, whose contents are protected and other applications (processes) cannot read this part or extract data to outside the enclave. Moreover, processes with higher privilege levels cannot access the data stored in enclaves.

Consequently, memory encryption is a good protection mechanism for the creation of the entropy value against attacks,[35] so that even the group management signaling server has no knowledge of the entropy value sent. This type of technology is used by vendors such as Intel Software Guard Extensions (SGX),[36] AMD Memory Encryption Technology,[37] and so on. Data transmission between the RTMCC signaling server and an enclave can be done using software techniques provided for the enclaves.[38] Since encrypted memory pages are controlled via the OS, OS tightening[39] can also support secure operations for data transfer between RTMCC and enclave.

4.4 | Computation process in clients

Clients must extract the entropy value from the notification they receive. This entropy value is used to generate a new key that is completely different from the old key. For this purpose, the entropy value and the old key are concatenated using a cryptographic key derivation function (KDF). The result of the KDF is passed through a one-way function similar to the hash functions method. In this way, the new SSK is generated by the client.

During the operation performed with the clients in the proposed *Entropy Service*, the outgoing client that does not have the entropy value cannot generate a new SSK. Therefore, the communication and computation complexity values in our proposed *Entropy Service* method are computed as (1) both when a client joins and when it leaves. Note that all the clients perform computations in parallel and moreover no new messages can be received unless they are notifications from the signaling server. Therefore, the number of notification messages with the proposed method is n. However, these messages are not newly created messages but messages already sent by the signaling server in which entropy values are added. This no new complexity is added to the system. Since the signaling server creates these messages once, the complexity is consequently (1).

4.5 | Synchronization

It may happen that two or more users join or leave the GC very soon. In this case, the signaling server sends separate messages to the users for each joining/leaving event. If this situation occurs repeatedly, there is a possibility that the synchronization of the clients will be disrupted. In other words, the clients need to be informed about which entropy value should be processed at that moment. For this purpose, the periodic keep-alive messages that the signaling server has already sent can be used to determine the existence of the clients. Keep-alive messages are sent to clients at much shorter intervals than notification messages. By adding a descriptive number for the current entropy value in these keep-alive messages, it is possible to publish which entropy value number is currently valid for the computation process of all clients.

4.6 | Client muting functionality

A client may perform many join and leave operations concurrently. The reason may be that the client has a problem with the network connection or with its own terminal. In this case, however, too many SSK changes are made. To prevent this, it is necessary to mute this client. If the client is muted, it will not be able to connect to GC. Moreover, if this client is ignored to trigger the SSK change, the security measures are weakened, which is not desirable. In this case, if it is found that a client frequently joins and leaves the GC, the system can be switched to the time-triggered method for a period of time.

Finally, in Algorithm 1, we provide the pseudocode for the overall process of the proposed method.

Algorithm 1 Pseudo-code of the proposed Entropy Service method.

```
Initialization:
   $Gen(Priv, Pub)$ ;                         // clients generate private and public keys for DTLS.
 1 $SSK_0 \leftarrow Gen(SSK)$ ;              // GC initiator (master client) generates the initial $SSK_0$
 2 $Dist(Pub)$ ;                              // distribute the public keys to clients
 3 $Synch \leftarrow 0$ ;                     // server's synchronization check for join/leave messages.
 4 $counter \leftarrow 0, Entropy \leftarrow 0$ ;   // Initialize synch. counter & entropy value.
 5 $t \leftarrow n, threshold \leftarrow x$ ;  // set triggering interval & frequent join threshold.
 6 ENTROPY Service:
    while (GC) do
 7      if $freq\_join < threshold$ then
 8          if $Synch == 1$ then
 9              $SERVER()$
10          else
11              $sleep$ t ;                   // wait for t time.
12 SERVER(): ;                                // server side operation.
13     $Entropy \leftarrow Gen(Entropy)$ ;    // generate new entropy value
14     $Message \leftarrow Gen(Message, Entropy)$ ;   // create the message with entropy value embedded.
15     $Dist(Message)$ ;                      // distribute notifications.
16     $CHANGE\_SSK(Clients, Message, Counter)$ ;   // clients perform SSK changing
17     $Counter \leftarrow Counter + 1$ ;     // synchronization counter increased for generated entropy values
18 CHANGE_SSK(Clients, Message, Counter): ;   // client side operation
19     foreach $client \in Clients$ do
20         $Entropy \leftarrow Ext_{Entropy}(Message(client))$ ;   // extract the entropy value
21         $Result = SSK_{Counter} || Entropy$ ;   // concatenation of entropy with $SSK_{Counter}$.
22         $SSK_{Counter} = Hash(Result)$ ;   // hash the result.
```

5 | EXPERIMENTAL ANALYSIS

This section provides details about the emulation environment and the analysis of the results. Since our proposal is event-driven, we made comparisons with the event-driven methods mentioned earlier (namely, key refreshing and hash functions methods) in our experimental tests. Note that the evaluation of the key hierarchy is excluded due to its lesser use in industrial applications compared to other methods.

5.1 | Details of the emulation setup

We emulated each method using mpi4py[40] on a 48-core Ubuntu 20.04 system. We use parallel computing programming on this system. We use a predefined core as a signaling server. Each remaining core is positioned as a separate client. Therefore, the implementation is done so that only the transactions associated with each client are executed in the client's dedicated core. All message traffic ((between the signaling server and clients) or (between clients)) in this system is actually messages between cores. A single core is specified as an actor client that joins or leaves the GC depending on the scenario.

The following is a summary of the steps taken for the test:

1. **Initialization** (a) Public Key Generation (b) Public Key Distribution (c) Symmetric Key Generation (d) Synchronization
2. **Operation**
3. **Finalization** (a) Synchronization (b) Assertion

The *Initialization and Finalization steps* are the same for each method being compared. The *Operation steps* are performed separately depending on the content of each method (key refreshing, hash functions, and entropy service). The SSK shared by each method is a 256-bit key. For hash functions and our proposed entropy service methods, clients perform a hash operation to generate the new SSK. We used the SHA-256 hash function for these operations. For the concatenation function of the entropy service, we used Password-Based Key Derivation Function2 (PBKDF2)-Hash-based Message Authentication Code (HMAC)-SHA256 as KDF, one of the most commonly password hashing algorithms for deriving keys from a password.

In the *Initialization step*, the public and private keys are generated. Rivest-Shamir-Adleman (RSA) keys with a length of 2048 bits are generated for each client. Then, the public keys are distributed so that each client has the public key of every other participant. The private keys are client-specific. These public and private keys are used to share the SSK. Namely, we have synchronized each client so that all clients are ready to proceed with the Operation step.

In the *Operation step*, each client follows the SSK changing procedure specified by the methods (key refreshing, hash functions, entropy service). In the *Finalization step*, we ensure that each client completes the *Operation step* by resynchronizing the clients. We also ensure that each client has exactly the same SSK (except for the actor (or master) client when it leaves the GC).

For a fair evaluation, we only measure the metrics between the operation steps and the final synchronization steps. The metrics observed are the number of messages, idle time and total time. Idle time represents the average CPU idle time until the SSK change is completed for all clients involved in the GC session. The total time represents the average time taken to complete the SSK change between all clients, and the number of messages is the average number of messages exchanged during the SSK change. The number of messages is calculated by counting the message function calls for the client leave and join messages sent by the server and the key change messages sent by the client (usually the one that initiated the GC).

5.2 | Results

In our tests, we compared our entropy service method from Section 4 with the event-driven methods explained in Section 3. We tested the entropy service, the hash functions, and the key refreshing methods using the cases of joining and leaving clients. Note that the hash functions method cannot be used when a client leaves the GC. For this reason, we use the key refreshing method for hash functions (as in Google Duo) when a client leaves the GC. To show the performance of the different methods with the same number of clients, we emulated the hash functions method for clients joining with hash functions and for clients leaving the GC using the key refreshing method. All tests are run 100 times and the average of the results is calculated and shown in the graphs.

Figure 7 shows the average values of the convergence time for all clients after the SSK is changed while the number of clients participating in GC increases. It can be observed that the number of messages increases when the number of users in the system increases as expected. For example, in *Entropy Service*, if there are 5 users and 1 user joins, there are 5 join messages and 1 key information message, while if 1 user leaves, there are 4 leave messages. For *Key Refreshing* method, if there are 5 users and 1 joins, there are 5 join messages and 5 new key messages, while if 1 user leaves, there are 4 leave messages and 3 new key messages. For *Hash Functions*, if there are 5 users and 1 user joins, there are 5 join messages and 1 new currently in use key message in the GC session.

Figure 7 shows that the proposed *Entropy Service* can save 98.7% and 98.3% of the total time compared to *Key Refreshing* and *Hash Functions*, respectively, when the number of clients is 5 and an additional user leaves or joins GC. If we

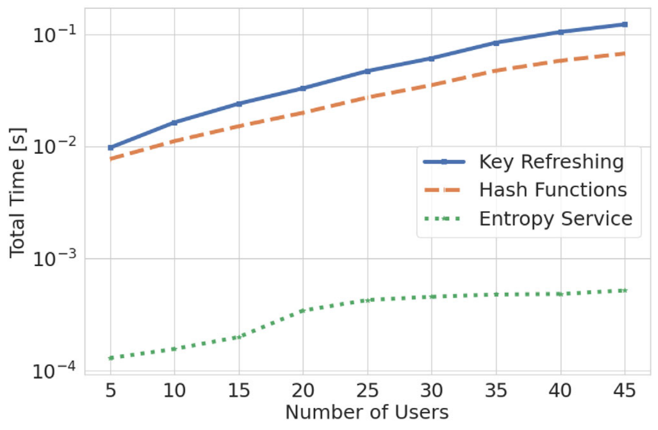

FIGURE 7 Comparison of the average total time for SSK change convergence over all users with the proposed Entropy Service method and event-driven methods

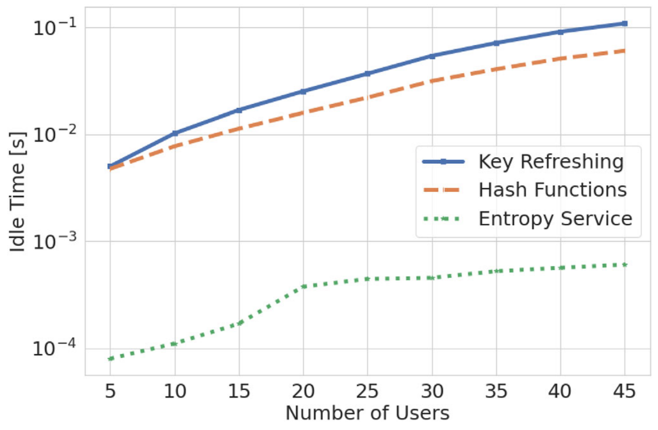

FIGURE 8 Comparisons of average idle time of all users with the proposed entropy service method and event-driven methods

increase the number of clients to 45, the time savings compared to the *Key Refreshing* and *Hash Functions* methods are 99.6% and 99.2%, respectively. Note that while both the *Hash Functions* and *Entropy Service* methods also use hashing, the *Hash Functions* method is slower than the proposed method. This is because *Hash Functions* require a large number of messages to change the SSK (which is also described in more detail in Figure 9). Although all methods use RSA-2048 encryption/decryption, the *Key Refreshing* method incurs a higher overhead due to the large number of message exchanges.

Figure 8 shows the idle time values for all clients after an SSK change, when the number of clients participating in GC increases. It can be observed that the idle time increases when the number of users in the system increases, as expected for all the considered methods. Figure 8 shows that the proposed *Entropy Service* can reduce the idle time by 98.4% and 98.3% compared to *Key Refreshing* and *Hash Functions*, respectively, when the number of clients is 5. When we increase the number of clients to 45, the idle time reductions are 99.4% and 98.99% compared to *Key Refreshing* and *Hash Functions* methods, respectively. The reason for the lower idle time in the proposed *Entropy Service* is that the entropy value is transmitted to the clients without any additional communication or computation overhead. Therefore, no notification messages are sent to the clients joining or leaving the GC, and hence no new SSK is created, which reduces the idle time.

Figure 9 shows the total number of messages for all clients after an SSK change when the number of clients participating in GC increases. It can be observed that the number of messages increases when the number of users in the system increases as expected for all the methods studied. Figure 9 shows that the proposed *Entropy Service* can reduce the number of messages by 64.7% and 53.8% compared to *Key Refreshing* and *Hash Functions*, respectively, when the number of clients is 5. When we increase the number of clients to 45, the reductions are 51.4% and 35.33% compared to *Key Refreshing* and *Hash Functions* methods, respectively. The improvement in messaging is achieved by having clients rely

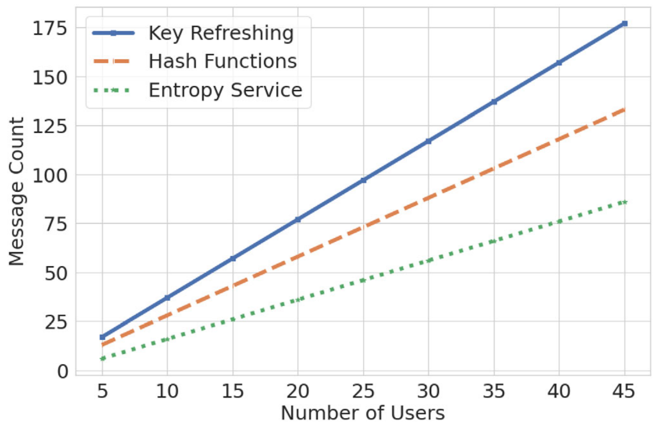

FIGURE 9 Overall number of message comparisons of the proposed entropy service method with event-driven methods

on a master client that distributes (the same) new key material to the many clients. This avoids (n^2) network communication complexity that would arise if every client negotiated new keys with every other client. Moreover, the main reason for the increased number of messages in *Hash Functions* is due to the fact that in this method, the newly created SSK is also forwarded to the joining new client by the client responsible for key distribution. On the other hand, the number of messages is the highest in *Key Refreshing* because the signaling server sends a join/leave message to all other clients to inform them that a new client has joined or left the session. In addition, a designated client must generate the new key and distribute it to others in case of join/leave events.

The experimental tests emulate an environment where there are no network delays and congestion. Even without network delays and congestion, the entropy service performs very well, both in terms of user CPU idle time and total convergence time. When network delays and congestion are also taken into account, the advantage is even greater than the *Key Refreshing* method. The main reason for the high advantage is that both the communication and computational complexity of the *Entropy Service* method is (1). We would also like to point out that the *Hash Functions* method cannot be used in cases when the client leaves. The test results also confirm our analysis. The performance improvement shown here ensures that no session, video frame, or audio is lost when SSK changes in a GC.

In summary, the above results show that the proposed *Entropy Service* can achieve significant performance gains in terms of total time, idle time, and number of messages compared to *key refreshing* especially a large number of clients are involved in RTMCC GC. Compared to the *Hash Functions* method, the gains are relatively smaller, but the *Entropy Service* still performs the best among the two event-driven methods.

5.3 | Discussions

In the *key refresh* method, a client generates the SSK. However, the computing power of a signaling server in RTMCC communication can be considered much higher than that of a client. Therefore, this high computing power of the server enables pure random number generation, which adds important layer of security to the system. Considering that reliability is also a security requirement, it is clear that the faster SSK change with *Entropy Service* is more reliable and does not allow frame/packet or session loss in a GC. This is due to the smaller number of messages resulting in a lower probability of frame/packet loss and mitigating synchronization issues (by embedding a descriptive number for the current entropy value in the periodic keep-alive messages). Note also that in practical applications, frame loss after key refreshing can be prevented if the new SSK is not used immediately (as in Google Duo's hash functions method and the proposed entropy service method).

The results in Figures 7 and 8 show that after reaching a certain number of users (about 30 users), the increase in idle time and total time for join/leave of a client for the proposed *Entropy Service* is no longer significant. These results show that as the number of clients increases, a saturation point tends to be reached.

Moreover, in the proposed *Entropy Service* method, the signaling server knows only one entropy value and does not have the information of the previous SSK, so the E2E security is not violated. The *Entropy Service* solution has good performance metrics, but if the entropy server sends the entropy to the client leaving the system the client can still generate the next key. Thus, the security weakness in the proposed method occurs only when the client and the signaling server cooperate. In this case, the user leaving the GC must penetrate into the signaling server and seize the entropy value. However, the same weakness applies to the key refresh and hash functions methods. This is because the client leaving the GC can attack another client and obtain the SSK directly. However, cooperation between the clients (which is sufficient to break *Key Refreshing* and *Hash Functions* methods) is more likely than cooperation between the server and the client (which is sufficient to break the proposed *Entropy Service* method). On the other hand, other schemes have indeed been proposed to achieve "lawful access" (see, for example¶). Thus, the threat is quite plausible and cannot be simply waved away. Indeed, it is one of the greatest threats to any E2E encryption system. Thus, our proposal does not pose any greater security risk than existing methods. Moreover, considering that it is more difficult to break into a server than into a client terminal, we can say that our proposed method is more secure in this respect.

Communication between the group management signaling server and clients is secured with TLS. This prevents leaving clients from monitoring the notification messages and obtaining the entropy value sent to the clients. There is also a risk that two users will always agree with each other. The one who continues the GC session can record the rest of the conversations and forward them to the client who left the GC. However, this is also a situation that cannot be prevented anyway during SSK changing.[41]

Attack models and key risks in proposed GKM: There are many attack models, depending on the specific capabilities of the attackers and where they can be placed, for example, in the server or in the client. We discuss semi-formal attack models in the following four main categories[42]:

i *Known-ciphertext attacks* are attacks that aim to obtain secret keys from known ciphertexts. The proposed system provides backward secrecy by changing the group's session keys after client joining/leaving. Thus, the known-ciphertext is no longer valid after another joining/leaving. The chosen hash function must be secure against collisions and collision attacks.
ii *Known-message attacks* are implemented by using existing message-tag pairs to compute secret keys. The hash algorithm must be secure against birthday attacks in a GC. The birthday attack is a type of brute force attack that exploits the mathematics behind the birthday problem in probability theory. The success of this attack depends largely on the higher probability of collisions between random attack attempts and a fixed degree of permutations.
iii *Chosen-message attacks* are used to eavesdrop on encrypted messages by instructing legitimate users to transmit specific messages. Pre-image attacks on cryptographic hash functions of the proposed system attempt to find a message that has a specific hash value. The cryptographic hash function used in GC should also resist attacks on its pre-image, that is, the set of possible inputs.
iv *Chosen-ciphertext attacks* decrypt ciphertexts by instructing legitimate users to faithfully decrypt the ciphertext queried by adversaries. As described in Reference 43, suitable hash functions are defined as secure against adaptive chosen ciphertext attack if their advantage to any efficient adversary is negligible.

There are other possible attack models in our semi-formal analysis as well. These other common security threats in the network and transport layers are:

- *Distinguishing attacks* aim at finding non-random behaviors of the algorithm. Since symmetric encryption is used for real-time communication in the system, the nature of permutation operations in symmetric encryption operations makes this system immune to these types of attacks.
- *Related-key attacks* aim to extract related master keys by finding linear relationships between them. Stream ciphers that are no longer used in industry, such as Wired Equivalent Privacy (WEP) and KASUMI, can be broken by these types of attacks. Modern stream ciphers do not have these types of vulnerabilities. In addition, post-quantum cryptography[44] can increase immunity to these attacks.
- *Side-channel attacks* aim to obtain physical information (eg, timing, power consumption, and electrical leaks in circuit). These attacks can infer data from GC in a temporal manner and can cause disruption (eg, latency and jitter) to a time-limited MC service in a shared infrastructure. The risk can be minimized by design and implementation on the hardware of the MC device that takes side channels into account.
- *Resource Hijacking* aims to abuse shared resource quotas to disrupt a single MC service by having services request more resources from the shared pool, thus compromising the availability of the service due to lack of resources. These

types of vulnerabilities originate from the endpoint software. Therefore, the use of Mobile Device Management (MDM) solutions can prevent these attacks.

- *Distributed Denial-of-Service (DDoS) attacks* aim to deny MC service to valid users, resulting in excessive distributed and synchronized network access. These attacks are possible in the areas of network management, GC isolation, and end-user node security. Poorly functioning infrastructure can compromise the availability and confidentiality of RTMCC services. On the other hand, recent advances in Artificial Intelligence (AI) and high computing power can help develop adversarial defenses.
- *Security vulnerabilities on some* end-devices (hardware/firmware/software) can introduce malware (computer worms and viruses) and zero-day exploits (software-based malicious behaviors) between devices in GC, which can inadvertently lead to a breach of confidentiality and unauthorized access.
- *An* impersonation attack is possible in the GC access authentication security domain. If an attacker captures a token or impersonates an element of the GC system, an authorization and authentication breach may occur. An attacker who obtains a stolen MC end-user device can perform this type of attack. The security of end-devices such as personal identification number (PIN) codes or remote emergency erase via MDM can be used against stolen or lost end-user devices.
- *Eavesdropping attacks* can occur when an attacker can monitor traffic (video or voice) and also understand the RTMCC configuration violating confidentiality. Mitigation actions and risk analysis have been previously defined in four main categories.
- *Tampering traffic* involves the malicious modification of data, which includes unauthorized changes to both persistent data and streaming data. This type of attack can always occur in a messaging system. The proposed system ensures that network traffic is always encrypted and complies with forward and backward secrecy.
- *Data leakage* happens due to activities performed by malicious users to gain access to GC information. This type of attack can be prevented with a secure end-device. The OS in the end-device of the MC service must be hardened OS. Moreover, the cryptographic keys must be stored in the Trusted Platform Module (TPM) chips inside the devices. The software must be executed in Trusted Execution Environment (TEE) areas.
- *Compromised/insider attacks* aim to compromise devices so that they can become potential surveillance devices for adversaries and unauthorized users. By gaining control of the GC patterns of the devices, that can control features such as the frequency of message triggering and synchronization between devices. These types of attacks can be mitigated through the use of hardware-based external provisioning devices such as crypto ignition keys (CIKs) or smart cards.
- *The potential threat of attacks from quantum computers* increases the requirements for advanced encryption/decryption techniques. Existing traditional algorithms based on traditional PKI are at risk of being attacked by quantum computers. Therefore, advances in quantum computing have accelerated the early adoption and testing of post quantum cryptography (PQC) algorithms. However, their existence in the network domain are still under investigation and there are many challenges to overcome.[44,45]

Finally, Table 3 summarizes the comparisons in terms of limitations, benefits and characteristics of each of the key change strategies discussed above, namely *Key Refreshing*, *Hash Functions*, *Key Hierarchy* and the proposed *Entropy Service* methods discussed in this paper.

6 | CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, we have explored all aspects of the possible methods that are suitable for a software implementation of SSK changing in E2E encryption of RTMCC. First, we present key changing procedures for E2E security of RTMCC systems in real-world application scenarios and the main reasons for requesting key changing in encrypted RTMCC sessions. We then examine the current and potential methods for changing RTMCC session keys and show under what circumstances the other existing methods perform key changes. Then, we propose how our proposal satisfies the need for fast SSK changing and is also easy to develop in terms of software implementation with a computational complexity of $\mathcal{O}(1)$. We then explain our entropy service solution with its implementation features such as synchronization and client muting.

Thanks to our test results and analysis, we showed that SSK change can be performed quickly with our solution. The performance of the proposed method in experimental tests shows that the proposed *Entropy Service* method can

TABLE 3 Comparisons of key change methods: key refreshing, hash functions and entropy service to provide RTMCC GC services

Key change strategy	Characteristics	Limitations	Advantages/benefits
Key refreshing strategy	• Predetermined client generates SSK. • Predetermined client distributes SSK to each client separately.	• Too many client messages for SSK distribution. • High convergence time due to message arrival latency. • Predetermined client performs too many calculations.	• Simplest and naive method • Requires no security knowledge for programmers. • Easy to implement
Hash functions strategy	• Each user knows the hash function in advance. • The previous SSK is passed through hash function	• Cannot be used for leaving clients. • High development complexity due to different implementation of user join & leave.	• Less messaging for • SSK distribution. • Less computation among clients overall.
Key hierarchy strategy	• A tree of keys is created. • The nodes of the tree contain key encryption keys • Session keys are distributed to leaf members using these keys.	• The entire system is executed in a stateful manner. • The states must be maintained seamlessly. • An additional key encapsulation operation is required. • The centre of tree is a single point of failure.	• Implementation via multicast messaging is relatively simple. • The centre of the tree does not rely on an entity for access control & key distribution. • Forward and backward secrecy can be easily implemented.
Entropy service strategy	• Signaling server performs Entropy Service. • Inserts entropy values in notification messages.	• Requires server-side development. • An additional concatenation operation is required.	• Can be used when a client leaves the GC. • Provides a fast SSK changing • Guarantees high GC quality without session & frame losses.

reduce 99.6% and 99.2% of the total time, 99.4% and 98.99% of the idle time and 51.4% and 35.33% of the number of messages compared to the *Key Refreshing* and *Hash Functions* methods respectively, when the number of users in the GC is 45. Although we take advantage of the non-standard signaling structure of RTMCC, our proposed solution is actually applicable to all RTMCC systems.

One of the important future directions in SSK changing is to standardize the process that RTMCC applications can use in real-world application scenarios. Another direction is to leverage recent developments in PQC algorithms to improve resilience of SSK change to future quantum computers.

ACKNOWLEDGMENTS

This work was partially funded by the National Program on Equipment and Scientific and Technical Infrastructure, EQC2018-005257-P under the European Regional Development Fund (FEDER).

CONFLICT OF INTEREST

On behalf of all authors, the corresponding author states that there is no conflict of interest.

DATA AVAILABILITY STATEMENT

Data not available: The datasets generated during and/or analyzed during the current study are not publicly available.

ENDNOTES

*https://www.5g-essence-h2020.eu/Home.aspx Accessed March, 2022

†https://www.darleneproject.eu/ Accessed March, 2022
‡https://respond-a-project.eu/ Accessed March, 2022
§https://www.cybersane-project.eu/ Accessed March, 2022
¶https://www.lawfareblog.com/evaluating-gchq-exceptional-access-proposal

ORCID

Engin Zeydan https://orcid.org/0000-0003-3329-0588

REFERENCES

1. Holmberg C, Hakansson S, Eriksson G. Web real-time communication use cases and requirements. IETF RFC 7478; 2015. https://tools.ietf.org/html/rfc7478
2. Rafaeli S, Hutchison D. A survey of key management for secure group communication. *ACM Comput Surv*. 2003;35(3):309-329. doi:10.1145/937503.937506
3. Thiyagarajan K, Lu R, El-Sankary K, Zhu H. Energy-aware encryption for securing video transmission in internet of multimedia things. *IEEE Trans Circuits Syst Video Technol*. 2019;29(3):610-624. doi:10.1109/TCSVT.2018.2808174
4. Go K, Lee I, Kang S, Kim M. Secure video transmission framework for battery-powered video devices. *IEEE Trans Dependable Secure Comput*. 2022;19(1):275-287. doi:10.1109/TDSC.2020.2980256
5. Mea DR. Key management for beyond 5G mobile small cells: a survey. *IEEE Access*. 2019;7:59200-59236.
6. Rams T, Pacyna P. A survey of group key distribution schemes with self-healing property. *IEEE Commun Surv Tutor*. 2013;15(2):820-842. doi:10.1109/SURV.2012.081712.00144
7. Vogt H, Awan ZH, Sezgin A. Secret-key generation: full-duplex versus half-duplex probing. *IEEE Trans Commun*. 2019;67(1):639-652. doi:10.1109/TCOMM.2018.2868714
8. Barnes R, Beurdouche B, Millican J, Omara E, Cohn-Gordon K, Robert R. The messaging layer security (MLS) protocol. Internet Engineering Task Force, Internet-Draft Draft-Ietfmls-Protocol-09; 2020.
9. Harney H, Muckenhirn C. Group key management protocol (GKMP) specification. IETF RFC 2093; 1997. https://tools.ietf.org/html/rfc2093
10. Barker E, Dang Q. Recommendation for key management—application-specific key management guidance. NIST Special Publication 800-57 Part 3 Revision 1; 2015.
11. Loreto S, Romano SP. How far are we from WebRTC-1.0? An update on standards and a look at What's next. *IEEE Commun Mag*. 2017;55(7):200-207. doi:10.1109/MCOM.2017.1600283
12. Nguyen DT, Nguyen KK, Cheriet M. NFV-based architecture for the interworking between WebRTC and IMS. *IEEE Trans Network Serv Manag*. 2018;15(4):1363-1377. doi:10.1109/TNSM.2018.2876697
13. Kirmizioglu RA, Tekalp AM. Multi-party WebRTC services using delay and bandwidth aware SDN-assisted IP multicasting of scalable video over 5G networks. *IEEE Trans Multimedia*. 2020;22(4):1005-1015. doi:10.1109/TMM.2019.2937170
14. 3rd Generation Partnership Project (3GPP). Mission critical services (MCS) group management; protocol specification. Technical Specification (TS 24.481) V17.4.0; 2022.
15. HeWea S. A scalable method of cryptographic key management for mission-criticalwireless ad-hoc networks. *IEEE Trans Inform Forensics Secur*. 2009;4(1):140-150. doi:10.1109/TIFS.2008.2009601
16. Ghosal A, Conti M. Key management systems for smart grid advanced metering infrastructure: a survey. *IEEE Commun Surv Tutor*. 2019;21(3):2831-2848. doi:10.1109/COMST.2019.2907650
17. De Cicco L, Carlucci G, Mascolo S. Congestion control for WebRTC: standardization status and open issues. *IEEE Commun Stand Mag*. 2017;1(2):22-27. doi:10.1109/MCOMSTD.2017.1700014
18. Omara E. Google Duo end-to-end encryption overview; 2020. https://bit.ly/3305HbL Accessed April 2021
19. Andrew E, Singh KM. Encrypting multiple party calls US Patent: US10666693B1; 2020. https://bit.ly/3xhUDRL Accessed April 2021
20. Signal. Signal messenger basics: voice or video calling; 2020. https://bit.ly/34AtxeC Accessed March 28, 2021
21. Jitsi. Does Jitsi support end-to-end encryption?; 2020. https://bit.ly/3GsKnti Accessed March 28, 2021
22. Janus. Plugin Demo: end-to-end encryption; 2021. https://janus.conf.meetecho.com/e2etest.html Accessed March 28, 2021
23. 3rd Generation Partnership Project (3GPP). Functional architecture and information flows to support Mission Critical Push To Talk (MCPTT). Technical Specification (TS 23.379) V17.9.0; 2021.
24. 3rd Generation Partnership Project (3GPP). Functional architecture and information flows to support Mission Critical Video (MCVideo). Technical Specification (TS 23.281) V17.6.0; 2021.
25. 3rd Generation Partnership Project (3GPP). Functional architecture and information flows to support Mission Critical Video (MCData). Technical Specification (TS 23.282) V17.9.0; 2021.
26. 3rd Generation Partnership Project (3GPP). Security of the Mission Critical (MC) service. Technical Specification (TS 33.180) V17.5.0; 2021.
27. Timmel P, Housley R, Turner S. NSA's cryptographic MessageSyntax (CMS) key management attributes. IETF RFC 7906; 2016. https://tools.ietf.org/html/rfc7906
28. Barker E. Recommendation for key management—general. NIST Special Publication 800-57 Part 3 Revision 5; 2020.
29. Vassilev A, Booth H, Staples R. Entropy as a service—NIST project; 2020. https://bit.ly/3xofJhi Accessed March 15, 2021

30. Garcia B, Gortazar F, Lopez-Fernandez L, Gallego M, Paris M. WebRTC testing: challenges and practical solutions. *IEEE Commun Stand Mag*. 2017;1(2):36-42. doi:10.1109/MCOMSTD.2017.1700005
31. Rescorla E. WebRTC security architecture. IETF RFC 8827; 2021. https://datatracker.ietf.org/doc/rfc8827/
32. Google WebRTC Team. Getting started with peer connections; 2020. https://webrtc.org/getting-started/peer-connections Accessed March 4, 2021
33. Alwen J, Coretti S, Dodis Y. The double ratchet: security notions, proofs, and modularization for the signal protocol. In: Ishai Y, Rijmen V, eds. *Advances in Cryptology - EUROCRYPT 2019. Lecture Notes in Computer Science*, Vol 11476. Cham: Springer; 2019.
34. Perrin T, Marlinspike M. The double ratchet algorithm; 2016. https://bit.ly/3sQP9tB Accessed April, 2021
35. Mofrad S, Zhang F, Lu S, Shi W. A comparison study of intel SGX and AMD memory encryption technology. Paper presented at: Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy; 2018: 1–8.
36. Intel. Software guard extensions; 2020. https://intel.ly/3dS8PJz Accessed March 28, 2021
37. Wu, Y., Liu Y, Tong J, et al. Comprehensive VM protection against untrusted hypervisor through retrofitted AMD memory encryption. Paper presented at: IEEE Symposium on High-Performance Computer Architecture; 2018: 441–453
38. Reed I. Intel SGX Intro: passing data between app and enclave; 2016. https://intel.ly/3sOjI31 Accessed April, 2021
39. Tevault D. *Mastering Linux Security and Hardening: Protect your Linux Systems from Intruders, Malware Attacks, and Other Cyber Threats*. Birmingham, England: Packt Publishing Ltd; 2020.
40. Dalcín L, Paz R, Storti M. MPI for Python. *J Parallel Distrib Comput*. 2005;65(9):1108-1115. doi:10.1016/j.jpdc.2005.03.010
41. Rescorla E. Security considerations for WebRTC. IETF RFC 8826; 2021. https://www.ietf.org/rfc/rfc8826.html/
42. Zhu B. Analysis and design of authentication and encryption algorithms for secure cloud systems. UWSpace; 2015.
43. Aea M. Comparison of hash function algorithms against attacks: a review. *Int J Adv Comput Sci Appl*. 2018;9(8):1-6.
44. Zeydan E, Turk Y, Aksoy B, Ozturk S. Recent advances in post-quantum cryptography for networks: a survey. 2022 Seventh International Conference On Mobile And Secure Services (MobiSecServ). *IEEE*. 2022;1-8.
45. Zeydan EA. Post-quantum era in V2X security: convergence of orchestration and parallel computation. *IEEE Commun Stand Mag*. 2022;6(1):76-82.

AUTHOR BIOGRAPHIES

Engin Zeydan received his Ph.D. degree in Electrical Engineering from Stevens Institute of Technology, Hoboken, NJ, USA, in 2011. He is currently a Senior Researcher at Centre Tecnologic de Telecomunicacions de Catalunya (CTTC). His research interests are in the areas of telecommunications and data engineering for networks.

Yekta Turk received his Ph.D. degree in Computer Engineering from Maltepe University, Istanbul, Turkey, in 2018. He is with Aselsan Corp. His research interests are in the areas of mobile radio telecommunications and computer networks.

Yaman Yagiz Tasbag received his B.Sc degree in Computer Engineering from Bilkent University, Ankara, Turkey, in 2020. He is currently pursuing his M.Sc degree from the same university. Previously, he was a Systems Engineer in Aselsan Corp. He is now a software development engineer in Amazon.