Detection And Countermeasure Scheme For Call-Disruption Attacks On SIP-Based Voip Services

Jea Tek Ryu[1], Byeong-hee Roh[2], Ki Yeol Ryu[2] and Myungchul Yoon[3]
[1]IP Service Team, Korea Institute of Patent Information, Seoul 146-8, Korea
[2]Dept. of Information and Computer Eng., Ajou University, Suwon 443-749, Korea
[3]Dept. of Electronics Engineering, Dankook University, Korea
[e-mail: ryujeatek@hotmail.com, : {bhroh, kryu}@ajou.ac.kr, myoon@dankook.ac.kr]
*Corresponding author: Ki Yeol Ryu

Received August 17, 2011; revised November 16, 2011; revised January 17, 2012; revised April 26, 2012; accepted June 19, 2012; published July 25, 2012

Abstract

Owing to its simplicity and flexibility, the session initiation protocol (SIP) has been widely adopted as a major session-management protocol for Internet telephony or Voice-over IP (VoIP) services. However, SIP has faced various types of security threats. Call-disruption attacks are some of the most severe threats they face, and can greatly inconvenience consumers. In this paper, we analyze such SIP call-disruption attacks, and propose a method for detecting and counteracting them by extending the SIP INFO method with authentication. Using the proposed method, both the target user and the SIP server can detect the existence of a call-disruption attack on a user and counteract the attack. We demonstrate the effectiveness of the proposed method from the viewpoint of computational complexity by configuring a test-bed with an Asterisk SIP proxy server and an SIP performance (SIPp) emulator.

Keywords: SIP, VoIP, Call-Disruption Attack, INFO Method, Authentication, Asterisk

This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency (NIPA-2012-(H0301-12-2003)).

http://dx.doi.org/10.3837/tiis.2012.07.008

1. Introduction

The session initiation protocol (SIP) is an application-layer signaling protocol used for establishing, maintaining, and terminating multimedia sessions [1]. Owing to its simplicity and flexibility, SIP has found widespread use as a session-management protocol for a variety of multimedia applications including Internet telephony, instant messaging, games, and IP multimedia subsystems (IMSs).

However, because of its similar structure with Hypertext Transfer Protocol (HTTP), including text-based message formats, SIP has faced various types of threats, such as distributed denial-of-service (DDoS), fuzzing, session hijacking, and call-disruption attacks [2]. Among these, call-disruption attacks are some of the most severe threats, greatly inconveniencing consumers. **Table 1** lists the typical types of SIP call-disruption attacks that use various SIP request messages. The detailed procedures of these attacks are provided in Section 3.

Table 1. Description of call-disruption attacks.

Attack Type	Attack Description
CANCEL	Cancel ongoing session setup requests using fake CANCEL messages
BYE	Terminate existing sessions using fake BYE messages
REGISTER	Remove or modify user registration information from the registration server using fake de-REGISTRATION messages
re-INVITE	Disrupt or intercept sessions under SIP-based mobility support environments using fake re-INVITE messages

Several research studies have dealt with such call-disruption attacks. Authentication- and encryption-based approaches [3][4][5][6] are ineffective, because SIP servers generate a certain amount of computational overhead for the encryption and decryption of individual messages. It was revealed in [7], [8], and [9] that the overhead generated by message authentication has a significant effect on the performance degradation in call setup delay. To deal with this, VoIP-specific Intrusion Detection System (IDS) architectures have been proposed [10][11]. However, these architectures require additional systems beside the server and user, as well as a very complicated structure to detect the various patterns of possible attacks. As in traditional IDSs, these IDS-based systems do not provide effective countermeasure mechanisms. While retransmission-based countermeasure schemes [12][13] use external systems to reduce the overhead, these methods may cause other call-disruption attacks.

In this paper, we propose an effective method for detecting and counteracting the types of call-disruption attacks listed in **Table 1**. The proposed method extends the SIP INFO method to detect a possible call-disruption attack on a target user, and notifies both the user and the server of the attack symptoms for a counteraction. The proposed method can be effectively applied to both static and mobile environments. The performance of the proposed mechanism is evaluated by configuring a test-bed with a Session Initiation Protocol performance (SIPp) [14] emulator and an Asterisk proxy server [15].

This paper is organized as follows. Section 2 describes some background issues related to the proposed scheme. Section 3 presents the call-disruption attack models considered in this paper, and Section 4 explains our proposed method for detecting and counteracting various call-disruption attacks as example usage scenarios. Next, Section 5 illustrates the effectiveness of the proposed method through a performance evaluation, and Section 6 concludes this paper.

2. Background

2.1 SIP Overview

SIP [1] is an application-layer protocol that enables multimedia sessions or calls to be set up, maintained, modified, or terminated. Similar to HTTP, SIP entities exchange text-based messages as request and response pairs. **Fig. 1** shows an example system architecture for SIP-based applications and services [16]. Each user agent (UA) registers with its domain's registrar server (1), and the registrar server then stores the information in its location server (2). The location servers store the location information of the UAs and determine where calls should be routed. The detailed usage of the location servers is illustrated in Section 3.2.B. UA(A) initiates a call request to UA(B) by sending an INVITE to its proxy server (3), and the proxy server resolves the location of UA(B) by consulting its location server (4). Next, A's proxy server transmits the request to B's proxy server (5), and the receiving proxy server then consults its location server (6) and forwards the request to UA(B) (7). After a three-way handshake between UA(A) and UA(B) in which 200 OK and ACK messages are exchanged, a media session between the two UAs is established (8).

An SIP message has a text-based format with the same three-part structure as that of an HTTP message: start line, message header, and message body. The start line identifies the message type and destination of the message. The message header includes signaling information, and the body contains additional information, e.g., information on the media used for the communication. The nature of this text-based message format makes it possible for attackers to form or alter the major attributes of SIP messages that can easily affect the call processing, which causes the call-disruption attacks listed in **Table 1**. With alterations of the attributes by an attacker, it is very difficult to differentiate fake requests from normal requests by legitimate UAs.

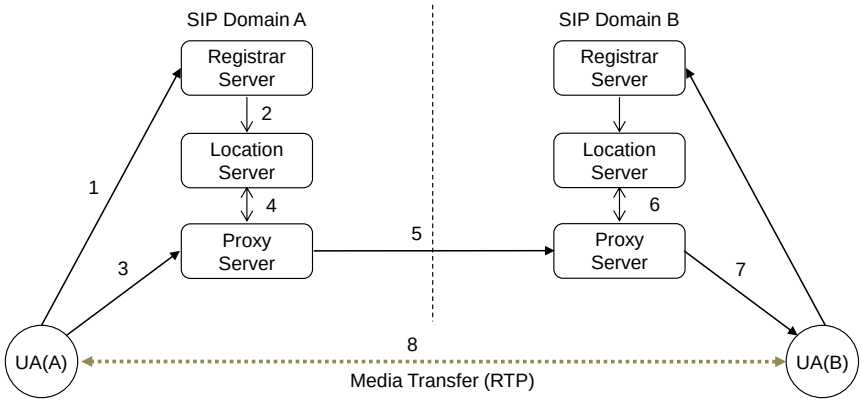


Fig. 1. Example system architecture for SIP-based applications [16]

2.2 SIP INFO Method

Various message forms and methods have been defined for SIP session control. The INFO method is used to carry optional application-level information on an SIP session [17]. The INFO method is not used to update the characteristics of an SIP dialog or session, but rather to allow applications to exchange information that might update their status.

To exchange information on a session, a UA sends an INFO request associated with an *InfoPackage*, or with the legacy INFO usage, for backward compatibility with the obsolete RFC 2976 [18]. An InfoPackage contains the content and semantics of the information carried in an INFO message. Optional INFO-based information on the session is included in the header and/or body of the INFO message. If an INFO request associated with an InfoPackage contains a message body, the body is identified by a `Content-Disposition` header field with an `Info-Package` value. The use of an InfoPackage associated with an INFO request for this proposed method is shown in **Fig. 7**.

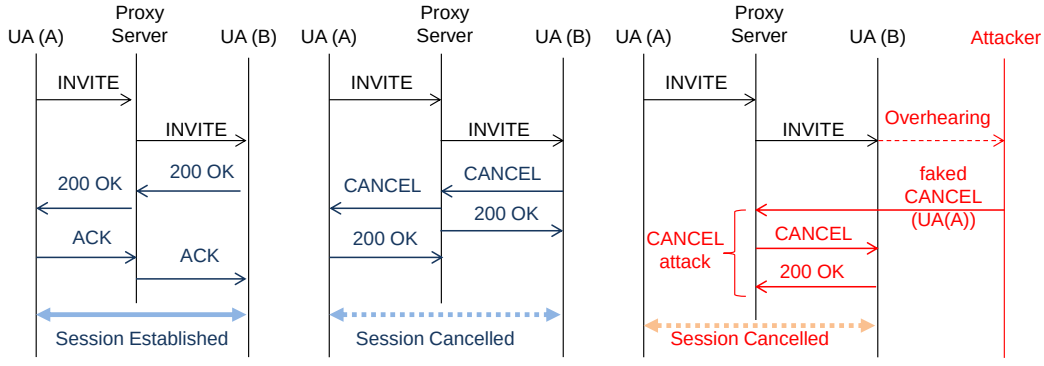
The UA receiving the INFO request replies with a 469 Bad InfoPackage response when it is unwilling to receive the INFO request. Otherwise, the UA must be prepared to receive the INFO request associated with the InfoPackage. If the INFO request is syntactically correct and well structured, the UA sends a 200 OK response. Otherwise, the UA sends an error response such as a Request Failure (4xx), Server Failure (5xx), or Global Failure (6xx) in accordance with the ordinary SIP error-handling procedures.

3. SIP Call-Disruption Attack Models

3.1 CANCEL and BYE Call-Disruption Attacks

The setup for an SIP session is conducted through a three-way handshake using INVITE, 200 OK, and ACK, as shown in **Fig. 2(a)**. A CANCEL message is used to cancel an ongoing session setup as shown in **Fig. 2(b)**. Using a CANCEL message, attackers can

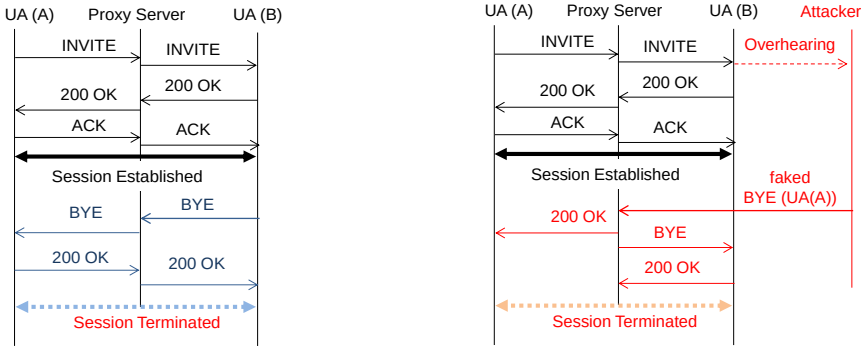
disrupt the session setup process as shown in **Fig. 2(c)**. In a CANCEL call-disruption attack scenario, UA(A) initiates a session setup by sending an INVITE message to UA(B). By eavesdropping on the INVITE, an attacker can generate a fake CANCEL message to terminate the setup process. For a CANCEL attack to succeed, the CANCEL message from the attacker has to be formulated before a 200 OK response from UA(B) is delivered to UA(A). The major attributes that should be included in CANCEL are the uniform resource indicators (URIs) of the caller and receiver and the Call-ID, which are identical to those included in the INVITE. Upon receiving a fake CANCEL message, the SIP server cancels the INVITE request from UA(A). Therefore, the UAs cannot complete a normal call setup.



(a) Normal session setup (b) Normal CANCEL (c) CANCEL attack

Fig. 2. Example of a CANCEL call-disruption attack

The established session can be terminated by sending a BYE message as shown in **Fig. 3(a)**. As in CANCEL attacks, an attacker eavesdrops on the three-way handshake session setup procedure. The attacker can then terminate an existing session by sending a fake BYE message with the proper session information obtained from the eavesdropped INVITE, as shown in **Fig. 3(b)**.



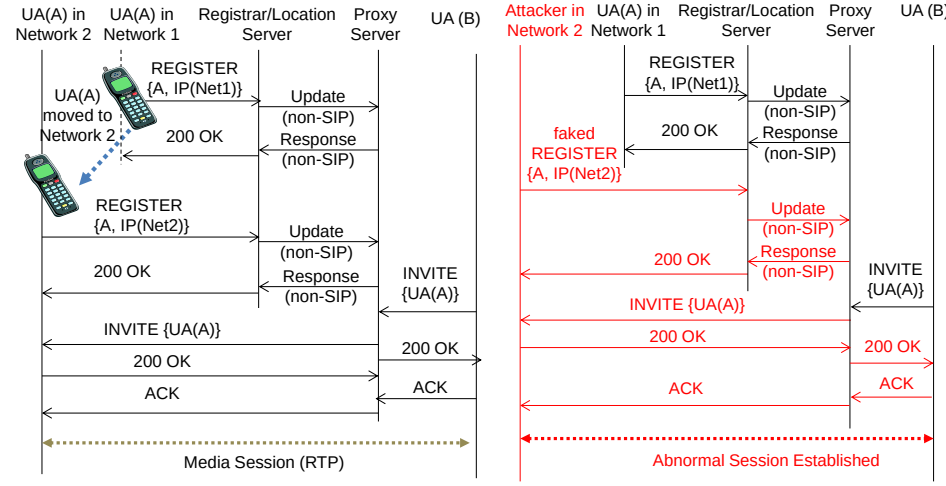
(a) Normal BYE (b) BYE attack

Fig. 3. Example of a BYE call-disruption attack

3.2 REGISTER Attack in Pre-call Mobility (or Roaming) Situations

SIP can support the mobility of a mobile host (MH) at the application layer [19]. Generally, SIP-based mobility is divided into two parts: pre-call mobility (or roaming) before a call, and mid-call mobility during a call. **Fig. 4(a)** shows a typical example of a normal pre-call mobility scenario. UA(A) initially registers with the registrar server by sending a REGISTER message including its contact information, and the server updates the record of the UA in the location server. The record is then reflected to the proxy server. After the UA moves to a different network, it re-registers and its records are updated at the servers. When UA(B) tries to establish a session with UA(A), it sends an INVITE destined for UA(A), and the proxy server then forwards the INVITE to the updated UA(A) location. As a result of this three-way handshake, a normal session between UA(A) and UA(B) is established.

Attackers can establish illegal sessions by utilizing a pre-call mobility mechanism. An example of an attack scenario is shown in **Fig. 4(b).** An attacker changes the records on UA(A) by sending a fake REGISTER including the attacker's location information. Since the registrar server cannot distinguish whether the message is issued from an attacker, the server changes the records on UA(A) with those provided by the attacker. A 200 OK message, which is a response for a successful change, is sent only to the attacker, but not to UA(A). Next, an INVITE from UA(B) to UA(A) is forwarded to the attacker, but not to UA(A). As a result of this process, an abnormal session between the attacker and UA(B) is established. A pre-call mobility attack has been classified by some researchers as a type of session hijacking attack. From the viewpoint of an attacked user, however, such an attack creates a situation in which normal calls cannot be made from or to the user. We therefore classify this type of attack as a call-disruption attack.

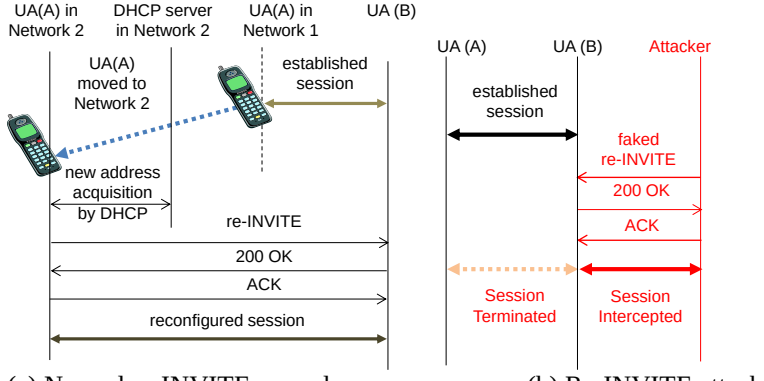


(a) Normal pre-call mobility (b) Pre-call mobility attack

Fig. 4. Example of a call-disruption attack on pre-call mobility

3.3 re-INVITE Attack during Mid-call Mobility

Once a session has been established using the three-way handshake INVITE/200 OK/ACK sequence, it can be modified by a re-INVITE, which is another INVITE/200 OK/ACK sequence. Using a re-INVITE, mid-call mobility can be supported without having to go through an intermediate SIP proxy, as shown in **Fig. 5(a)**. When moving from one network to another, UA(A) acquires a new IP address, and then sends a re-INVITE message to the corresponding UA(B), allowing the SIP session to continue with the new address. However, attackers can intercept the session by sending a fake re-INVITE message, as shown in **Fig. 13(b)**. Based on the SIP specifications, there is no way for UA(B) to distinguish whether the re-INVITE is from UA(A) or from an attacker.



(a) Normal re-INVITE procedure (b) Re-INVITE attack

Fig. 5. Example of a re-INVITE attack during mid-call mobility

4. Detection and Counteraction Scheme using an Extension of the INFO Method

4.1 Overall Architecture

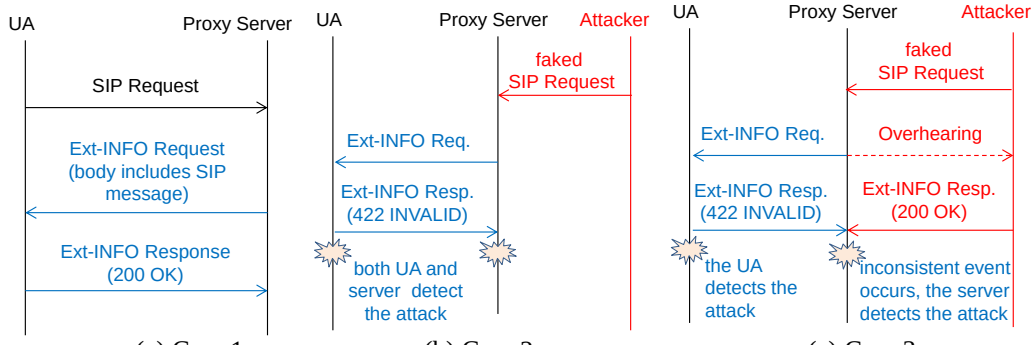
4.1.1 Simple extension of INFO Method (Ext-INFO)

To detect call-disruption attacks using an SIP request such as a CANCEL, BYE, de-REGISTRATION, or re-INVITE message, we propose the use of a scheme that utilizes a simple extension of the INFO method, or Ext-INFO, as shown in **Fig. 6**. When an SIP proxy server receives a request message related to a call-disruption attack, it sends an Ext-INFO request to the UA that sent the message. The body of the Ext-INFO request includes the entire message that the proxy server has received most recently from the UA. Note that the message may be sent from either the UA itself or an attacker. There are three use cases for Ext-INFO:

- *Case 1* (without an attack): The UA replies to an Ext-INFO response with a 200 OK message if the body in the Ext-INFO request is identical to the message most recently

sent to the server, as depicted in **Fig. 6(a)**. Note that the UA should store the latest message sent to the server for Ext-INFO requests. The server then accepts the SIP message and provides the appropriate service.

- *Case 2* (without a fake response by an attacker): If the body in the Ext-INFO request is not identical to the message it sent to the server most recently, the UA replies to the Ext-INFO response with a 422 INVALID message, which is shown in **Fig. 6(b)**. When the UA finds any differences between the body of the received Ext-INFO request and the latest message sent to the server, it can determine that it is the target of a call-disruption attack. Similarly, the server also discovers the attack by receiving the 422 INVALID message from the UA.
- *Case 3* (with a fake response by an attacker): When a server sends an Ext-INFO request, an attacker may eavesdrop on the request. The attacker may then send an Ext-INFO response with a 200 OK to complete the call disruption, as shown in **Fig. 6(c)**. However, the server can check for any conflicts between the two received messages, and if such a conflict exists, the server can easily recognize it as evidence of an attack.



(a) Case 1 (b) Case 2 (c) Case 3

Fig. 6. Basic procedure for detecting a call-disruption attack using Ext-INFO

An example of the Ext-INFO request format is shown in **Fig. 7**. As mentioned in Section 2.B, an INFO request may contain an InfoPackage carrying application-level information, including the content and semantics of the information to be carried. As shown in **Fig. 7**, the proposed Ext-INFO request is indicated in the `Info-Package` header field as EXT-INFO, with InfoPackage provided in the `Content-Disposition` header field.

In **Fig. 7**, it is assumed that a call setup between two user agents, UA(A) and UA(B), is progressing through a proxy server, and that a CANCEL message from UA(A) has been received at the server. As mentioned before, the CANCEL message might be sent from either the UA(A) itself or an attacker. As shown in **Fig. 7**, the Ext-INFO request consists of a header and body, as in any other SIP message. Unlike an ordinary INFO method, however, since the server initiates the request and wants to receive a response to it, its URI appears in the `From` field in the INFO header. In addition, since UA(A) has been recorded as the sender of the CANCEL message, the URI of UA(A) is designated to the recipient of the

Ext-INFO request, which appears in the `To` field in Ext-INFO header. The entire CANCEL message received by the server is included in the Ext-INFO body. The `Call-ID` should be identical to the call identifier of the session between the two UAs.

When UA(A) receives an Ext-INFO request, it checks whether the message included in Ext-INFO body is identical to its latest message sent to the server lastly. Based on the result of the check, either a 200 OK or a 422 INVALID response is sent to the server. Since the UA receiving an INFO request should reply with a 200 OK or other error-handling response, no additional overhead is incurred when using the Ext-INFO method as compared to a conventional INFO method.

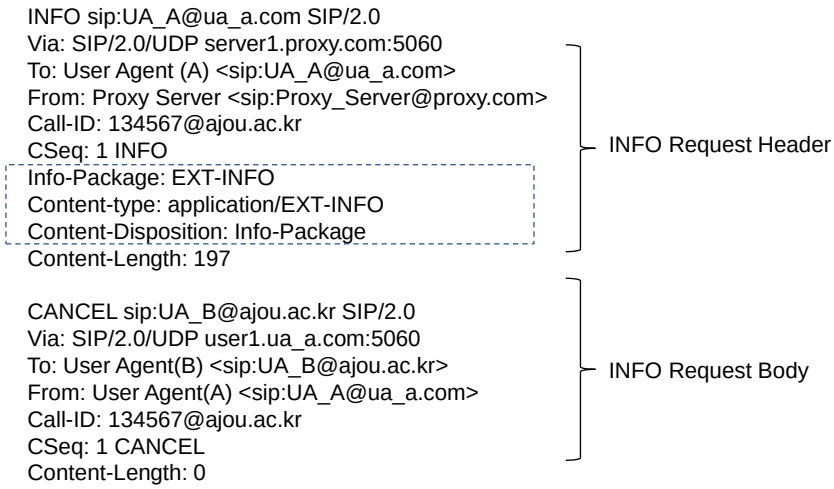


Fig. 7. An example of an extended INFO message format

For cases 2 and 3 in **Fig. 6**, the server can protect the session for the UA by ignoring the fake SIP message sent by the attacker. In case 3 in **Fig. 6(c)**, if the 200 OK message issued by the attacker reaches the SIP server earlier than the 422 INVALID message sent by a normal UA, the SIP server will accept the fake 200 OK message and fail to detect the attack. To avoid this situation, the proposed method uses the following waiting mechanism: an Ext-INFO Req. message loss may occur since SIP messages are generally sent over the User Datagram Protocol (UDP). In the case of a message loss, the SIP server resends the message according to the retransmission mechanism for non-INVITE request messages [21]. Although the server receives the first Ext-INFO response after sending the request, it delays its decision until the retransmission time for the requested message expires or until a second Ext-INFO response arrives. Since each response is generally received within the retransmission time interval during normal network conditions, the problem described above can be solved.

4.1.2 Hybrid of Extended INFO with Authenticated Registration Procedure

The Ext-INFO method has a severe weakness, as shown in **Fig. 8**. When the server receives

an SIP request from a legitimate UA, it sends an Ext-INFO request to the UA. If an attacker that knows the mechanism overhears the request, it can send a malformed response with a 422 INVALID response, which conflicts with the 200 OK response sent by the UA. The server may then decide that the received request is fake and ignore it, preventing a proper SIP service from being provided to the UA. This may cause another type of call-disruption attack.

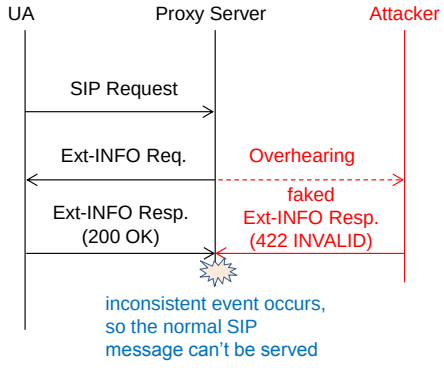


Fig. 8. Drawback of Ext-INFO

To overcome this drawback of the Ext-INFO method, we propose a hybrid method using an extended INFO with an SIP authentication mechanism, or Hyb-INFO. The SIP specification provides a user-authentication mechanism based on HTTP digest access authentication [16], which is a simple challenge-response protocol of authentication using a one-way function, e.g., MD5 or SHA1. There are two types of HTTP digest authentication mechanisms: user-to-user, and user-to-proxy [2]. In the former case, the authentication procedure takes place between two UAs, or between one UA and an SIP registration server. The latter case operates between a UA and an SIP proxy server for the call-setup processes. Based on the SIP-based service architecture shown in **Fig. 6**, let us consider a user-to-proxy authentication procedure. The proposed Hyb-INFO procedure is shown in **Fig. 9**. A Hyb-INFO request includes the body of the received SIP request message and a 407 Proxy Authentication Required message with a nonce value and other authentication algorithm information.

Upon receiving a Hyb-INFO request, the UA calculates the information based on the nonce and the algorithm information from the 407 Proxy Authentication Required header. The calculated information is included in Proxy-Authorization. In addition, the UA checks whether the message body included in the Hyb-INFO request is identical to its last message. Depending on the result, either a 200 OK or 422 INVALID message is formed, as shown in **Fig. 9(a)** and **Fig. 9(b)**, respectively. The UA then sends the Hyb-INFO response with a Proxy-Authorization and 200 OK (or 422 INVALID) message to the server.

Note that, while attackers may overhear the Hyb-INFO request of the server, they cannot generate a suitable Hyb-INFO response without access to the authentication information. The drawback of Ext-INFO can therefore be resolved.

After the UA or server detects any evidence of a call-disruption attack, all of the SIP messages exchanged between them can be encrypted using an optional process. Note that the exchanged messages are only encrypted during sessions in which an attack has been detected. That is, SIP messages exchanged between sessions with no evidence of an attack do not need to be encrypted. This can reduce the computational complexity compared to cases in which all messages are encrypted. Any encryption mechanism can be used for this optional process, but such a discussion is beyond the scope of this paper.

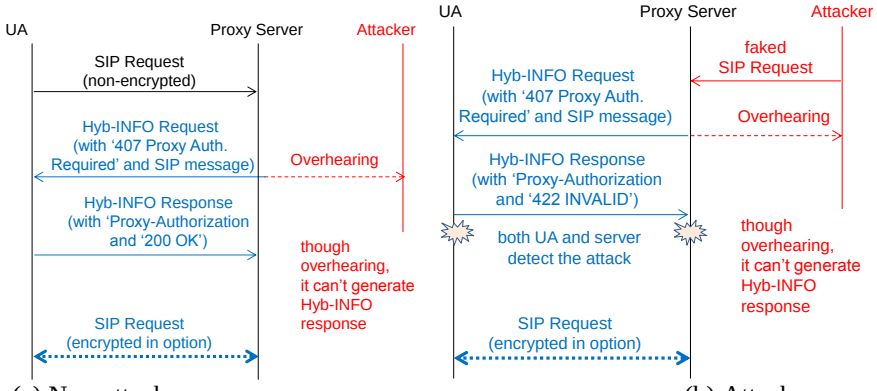


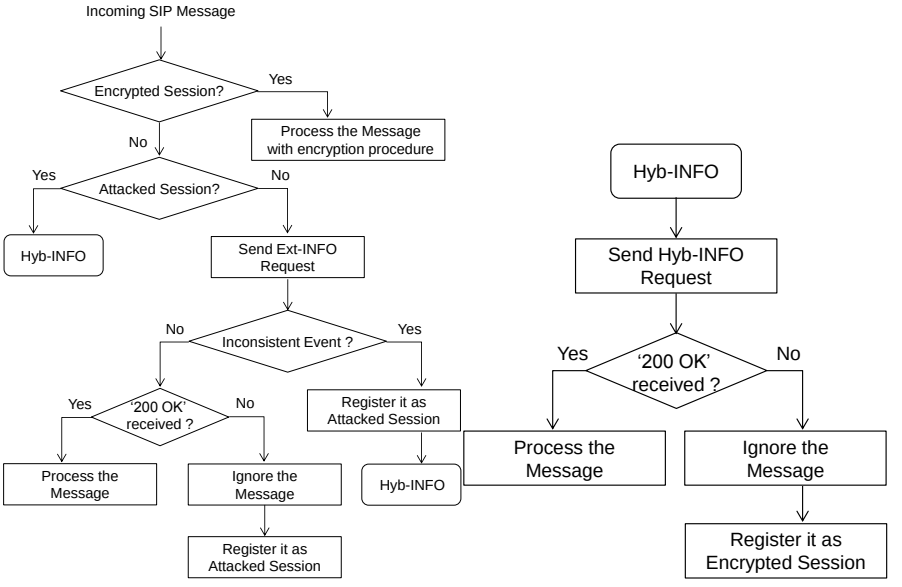
Fig. 9. Hyb-INFO procedure

4.1.3 Overall Procedure for the Detection of and Countermeasures against Call-Disruption Attacks

The overall proposed procedure, which combines Ext-INFO and Hyb-INFO, as used at the SIP proxy server, is shown in **Fig. 10(a)**. We define a session as the combination of three tuples <Call-ID, From, To>, which are included in SIP messages. Based on the definition of the session, the encrypted and attacked sessions are maintained. Since practical SIP proxy servers such as those described in [15] have a mechanism to maintain their sessions using information similar to the tuples, the procedure can be easily implemented by adding two bits that indicate whether the session has been attacked or is encrypted, as shown in **Fig. 10(a)**.

If an incoming SIP request is listed as an encrypted session, then the message is processed normally using a proper decryption process, since it is very difficult for attackers to intrude into such a session. An SIP session is represented by four fields: the source IP address, caller's URI, receiver's URI, and call-ID. Each session is set as non-encrypted upon first arrival. A session is determined as being encrypted using one of the resulting processes of Hyb-INFO, as shown in **Fig. 10(b)**. For incoming messages that are not listed as encrypted or attacked sessions, the Ext-INFO procedure is initiated. If the response to the Ext-INFO request is a conflict or a 422 INVALID message, the session is registered as an attacked session. On the other hand, for a 200 OK response, the message is properly serviced. Likewise, the proposed method reduces the computational complexity

significantly through the partial use of authentication or encryption processes only for severely attacked sessions, unlike traditional authentication- or encryption-based approaches, which use authentication or encryption/decryption processes for all messages.



(a) Overall procedure (b) Hyb-INFO procedure

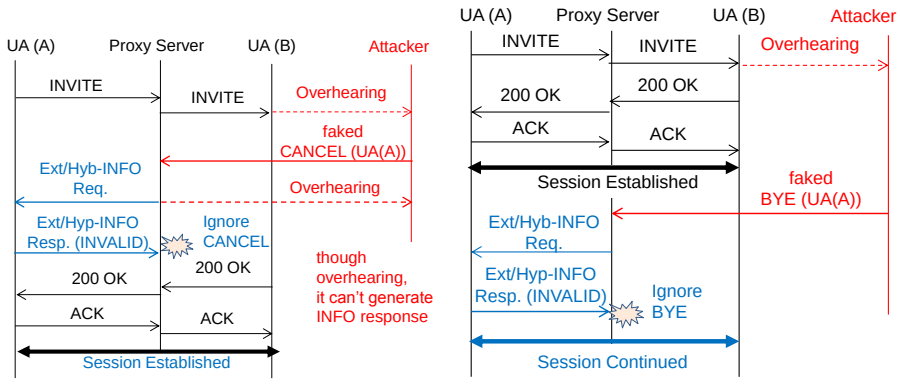
Fig. 10. Overall procedure of the proposed method

4.2 Example Usage Scenarios

4.2.1 Detection and Countermeasure of CANCEL and BYE Call-Disruption Attacks

Using the proposed method shown in **Fig. 10**, a CANCEL attack can be prevented, as shown in **Fig. 11 (a)**. A model of a CANCEL attack is given in **Fig. 2(c)** in Section 3.A. After the server receives a fake CANCEL message from an attacker, it generates an Ext-INFO request and sends it to UA(A). Even if the attacker overhears the message, it cannot generate a response to the Ext-INFO request if it does not know the shared key between the server and UA(A). UA(A) responds with a 422 INVALID message, since it did not send the SIP message. The server then acknowledges that the CANCEL message is invalid and ignores it. As a result, a 200 OK response to the INVITE is successfully delivered from UA(B) to UA(A). Finally, a media session between UA(A) and UA(B) is established.

As with CANCEL attacks, the server and a UA can easily detect an attacker's BYE message using the proposed method, and can therefore protect their session. The procedure for detecting and countering a BYE attack is shown in **Fig. 11(b)**.



(a) Protection from a CANCEL attack (b) Protection from a BYE attack

Fig. 11. Protection from CANCEL and BYE call-disruption attacks

4.2.2 Detection and Countermeasure of REGISTER (or Pre-call Mobility) Attack

Using the proposed method, the REGISTER attack illustrated in **Fig. 4(b)** in Section 3.B can be detected and blocked, as shown in **Fig. 12**. When the registrar server receives a REGISTER to update the UA's record, it sends an Ext-INFO request destined for both the previous and current UA locations without changing the record. Note that since the UA's record is not changed, the server knows both the previous and newly requested UA records, and can send an INFO request to both the previous and current UA locations. Since no call information is requested for the pre-call mobility procedure, only the use of the Hyb-INFO scheme can be considered. If the REGISTER is fake, that is, the UA's location has not changed, the legitimate UA provides a 422 INVALID response, to which the attacker cannot generate a proper response. Based on the UA's response, the server ignores the attacker's REGISTER and does not change the UA's record. Normal sessions can then be established between this UA and others. Note that even if the attacker sends a 200 OK response, the server can detect a conflict and invoke appropriate countermeasures.

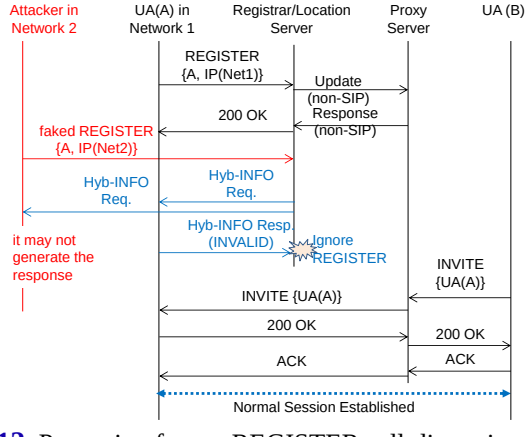


Fig. 12. Protection from a REGISTER call-disruption attack

4.2.3 Detection and Countermeasure of re-INVITE Attack during Mid-call Mobility

A re-INVITE attack can be detected using the proposed method, as shown in **Fig. 13**. When UA(B) receives a re-INVITE message with a change in location information, it sends a Hyb-INFO request to both the previous and currently indicated locations. Note that, since the re-INVITE process is conducted between the UAs directly, we can only use the Hyb-INFO scheme. If a re-INVITE comes from an attacker, UA(A) sends a Hyb-INFO response with a 422 INVALID message. With this response, UA(B) ignores the re-INVITE, and the correct session with UA(A) can continue without disruption.

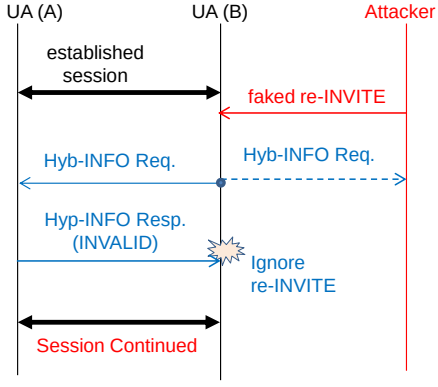


Fig. 13. Protection from a re-INVITE call-disruption attack

5. Performance Evaluation

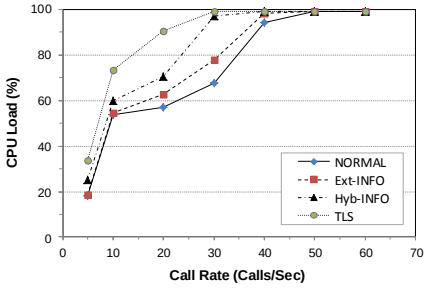
To evaluate the effectiveness of the proposed scheme, we constructed the following test-bed. The test-bed consists of two PCs for both normal and attack SIP message generation, and one SIP proxy server, each of which are connected using a 100 Mbps fast Ethernet link. For normal SIP call generation, we used an emulation PC with a 2.5 GHz Intel Core Duo CPU and 2 GB of RAM, running open VoIP emulation software called SIPp [14], which can not only make virtual SIP calls but can also measure the system performance. For attack message generation, we used another PC with the same specifications used in the normal call generating PC. We implemented a program to capture normal INVITE messages and generate fake call-disruption request messages on the attack PC using the *libpcap* library. We configured an Asterisk SIP proxy server [15] on a PC with a 2.66 GHz Intel Q6600 CPU and 4 GB of RAM, running Linux with an Ubuntu 10.4 kernel. Although the CPU has dual cores, we only considered one processor for the experiments, as described in Article (CrossRef Link).

[**20**], because SIP performance under multiple processors or multi-cores is beyond the scope of this paper. The proposed schemes, Ext-INFO and Hyb-INFO, were implemented on the normal SIP message generating and SIP proxy server PCs. A Transport Layer Security (TLS)-based encryption scheme [7] provided by SIPp was configured on these

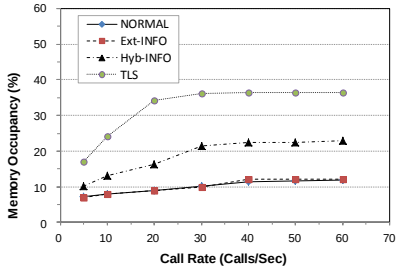
systems for performance comparisons. In the TLS-based scheme, since all SIP messages are encrypted, attackers cannot disrupt any sessions without knowing the secret keys shared between the server and the UAs. However, the UAs and server should decrypt and encrypt all SIP messages exchanged between them.

We have seen that the proposed method exactly detects the call-disruption attacks for all the generated calls, in which we made the attacks according to attack models described in Section 3.2. In our experiments, no false-positive events were found, i.e., 100% of the attack trials were detected and fixed.

To show the effectiveness of the proposed scheme from the viewpoint of computational complexity, we carried out the following experiments. We varied the average SIP call generation rates (5, 10, 20, 30, …, and 60 in calls/s), and set the average call duration to 30 s with an exponential distribution. To reflect the practical use of SIP server resources, such as CPU and memory, we considered the whole life of a call, from its session setup to its termination. That is, after the duration exponentially given for a call is expired, the call sends a BYE message to terminate its session. Note that resources are utilized during a call and at the session setup and termination. However, since Real-time Transport Protocol (RTP) packets for a conversation are exchanged through a separate connection between terminals, but not through the SIP server, they cannot affect the usage of the SIP server's resources. **Fig. 14** shows performance comparisons for the CPU load and memory occupation. In **Fig. 14**, the original SIP mechanism without authentication is marked as NORMAL. As shown in **Fig. 14 (a)**, the CPU load for each scheme increases as the average call rate increases. However, the degree of increase for TLS is considerably higher than that of the other schemes. When the call rate increases to greater than 30 calls/s, the CPU load for TLS reaches 100%, which means that no SIP messages can be served. The CPU loads for Hyb-INFO are larger than those for Ext-INFO as Hyb-INFO requires an additional authentication process compared to Ext-INFO. The NORMAL mechanism shows the lowest amount of CPU load. **Fig. 14 (b)**, shows that the amount of memory required for processing SIP messages increases with increasing average call rate. The memory occupancy of Ext-INFO is almost the same as that of NORMAL, and is considerably lower than that of Hyb-INFO and TLS. TLS shows the largest memory occupancy. The results of **Fig. 14** indicate that the proposed scheme can protect users from call-disruption attacks with much lower system utilization compared to TLS.



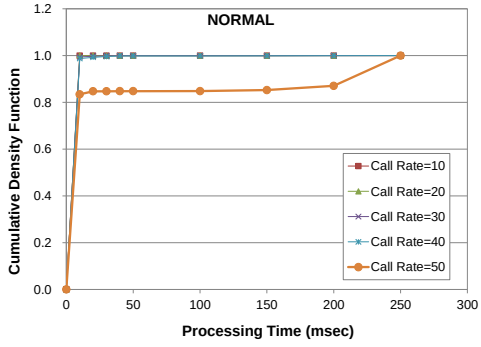
(a) CPU load



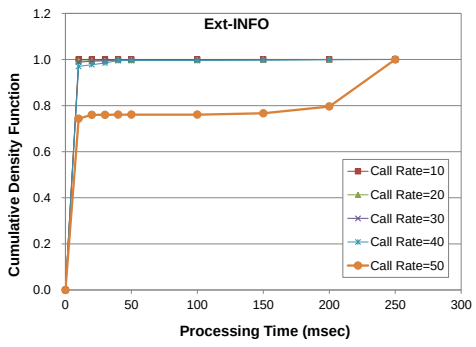
(b) Memory usage

Fig. 14. Performance in terms of CPU and memory load

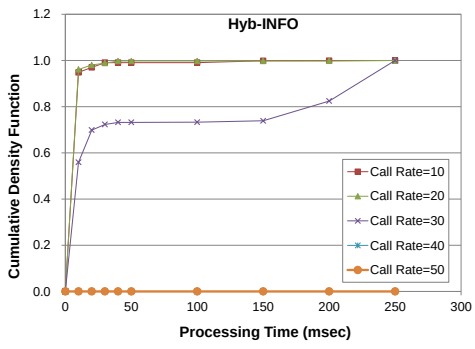
In **Fig. 15**, the average processing times as cumulative density functions (CDFs) are compared for NORMAL, Ext-INFO, Hyb-INFO, and TLS for various call rates of 10, 20, 30, 40, and 50 calls/s. We define the processing time for an SIP message as the time spent between its arrival at and its departure from the SIP server. The processing time includes the queuing (or buffering) delay, decryption/encryption delay for authenticated messages, attack decision delay, and transmission delay. The queuing delay is the time that a message spends in the waiting room until the service is initiated. The message should be decrypted, analyzed, and encrypted during the service time if it is authenticated. The service time increases with increasing number of authenticated messages. As a result, the queuing delays and processing times for individual messages will increase as well.



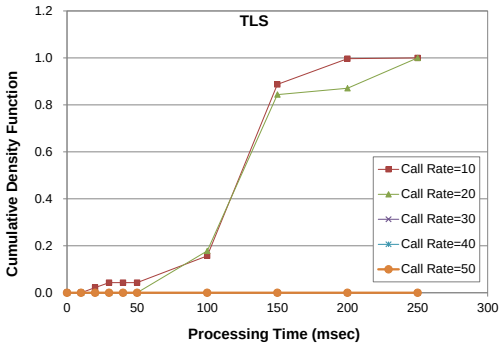
(a) NORMAL



(b) Ext-INFO



(c) Hyb-INFO



(d) TLS

Fig. 15. Average processing times (in milliseconds)

As shown in **Fig. 15(a)** and **Fig. 15(b)**, when the call rate is below 40 call/s, the processing times of NORMAL and Ext-INFO are consistently less than 10 ms. When the call rate is 50

calls/s, they are increased up to 250 ms. In addition, Ext-INFO shows longer processing times than NORMAL. Note that messages are still served even when the processing times are increased up to 250 ms. However, the processing times for Hyb-INFO are extremely long when the call rate is 40 calls/s or more, as can be seen from **Fig. 15 (c)**, and messages can therefore not be served. An extremely long message processing delay creates a type of DoS (Denial of Service) at the server. Even with call rates of below 30 calls/s, the processing delays are much longer than in NORMAL and Ext-INFO. As shown in **Fig. 15(d)**, the processing time of TLS is much longer than that of the other schemes even when the call rate is below 20 calls/s. When the call rate is 30 calls/s or more, the processing times become extremely long, and the server falls into a DoS state.

As mentioned earlier, the common methods for counteracting against SIP call-disruption attacks utilize encryption-based approaches [3][4][5][6]. However, encryption-based schemes require very high computational complexity and a large memory space, as shown in **Fig. 14** and **Fig. 15**. Compared with encryption-based schemes, the proposed method requires much lower system requirements. Note that the performance of the proposed method reflects the worst case in which attackers generate fake call-disruption requests for all normal SIP sessions, while the performance of the TLS-based scheme is based on an ordinary situation since all SIP messages should be encrypted when using this scheme. In normal operational environments of SIP-based services, it is very difficult for attackers to generate call-disruption requests for all SIP sessions because attackers should capture packets from the UAs to generate a successful attack; however, the UAs are located on different and distributed networks.

6. Conclusion

In this paper, we proposed an extended INFO-based detection and countermeasure scheme for various types of SIP call-disruption attacks. Some example scenarios of such attacks and the corresponding countermeasure procedures using the proposed scheme were illustrated. The performance of the proposed scheme was compared to the TLS-based scheme, which is a widely used scheme for providing countermeasures against SIP call-disruption attacks. Based on our experiments, we determined that the proposed method detects call-disruption attacks precisely. In addition, we demonstrated that the proposed method can work with much lower system requirements (lower CPU load, memory occupation, and processing delay) than a TLS-based scheme under an environment of various types of call-disruption attacks.

Voice-over IP (VoIP) is a technology that provides traditional telephone services on Internet Protocol (IP) networks. A call-disruption attack is one of the most severe types of threats, and can cause great inconvenience to VoIP users, severely degrading their quality of experience. If VoIP users are frequently targeted by attacks and they complain about the inconvenience caused by them, VoIP service providers may have to construct a defense mechanism for attack prevention. The goal of our proposed method is to protect both users and service providers from call-disruption attacks while keeping the need for additional memory and processing overhead at a minimum. Based on our experimental

results, which show that the proposed method can achieve its security goal very effectively as compared to existing solutions, we expect that providers may adopt the proposed method as a good candidate solution against call-disruption attacks, despite extra costs such as the storage of the last messages sent to the UAs and an incurred delay owing to a required interaction between the UAs and servers.

References

[1] J.Rosenberg, H. Schulzrinne, G. Cvamarillo, A. Johnston, J. Peterson, R. Spark, M. Handley, and E. Schooler, "SIP : Session Initiation Protocol," IETF RFC 3261, June 2002.

[2] D. Sisalem, J. Floroiu, J. Kuthan, U. Abend, and H. Schulzrinne, *SIP Security*, John Wiley & Sons Ltd., 2009.

[3] D. Geneiatakis, and C. Lambrinoudakis, "A Lightweight Protection Mechanism against Signaling Attacks in a SIP-Based VoIP Environment," Telecommunication Systems, Vol.36, No.4, pp.153-159, Dec. 2007. Article (CrossRef Link).

[4] A. Bremler-Barr, R. Halachmi-Bekel, and J. Kangasharju, "Unregister attacks in SIP," IEEE 2nd Workshop on Secure Network Protocols'2006, Nov. 2006. Article (CrossRef Link).

[5] F. Wang, and Y. Zhang, "A New Provably Secure Authentication and Key Agreement Mechanism for SIP Using Certificateless Public-Key Cryptography," Computer Communications, Vol.31, No.10, pp.2142-2149, June 2008. Article (CrossRef Link).

[6] H. Takahara, and M. Nakamura, "Enhancement of SIP Signaling for Integrity Verification," IEEE/IPSJ SAINT'2010, Jul. 2010. Article (CrossRef Link).

[7] S. Salsano, L. Veltri, D. Papalilo, "SIP Security Issues: The SIP Authentication Procedure and Its Processing Load," IEEE Network Magazine, Vol.16, No.6, pp.38-44, Nov/Dec 2002. Article (CrossRef Link).

[8] E. Cha, H. Choi, and S. Cho, "Evaluation of Security Protocols for the Session Initiation Protocol," IEEE ICCCN'2007, Aug. 2007. Article (CrossRef Link).

[9] S. V. Subramanian, and R. Dutta, "Comparative Study of Secure vs. Non-secure Transport Protocols on the SIP Proxy Server Performance: An Experimental Approach," IEEE ARTCom'2010, Oct. 2010. Article (CrossRef Link)

[10] Y. Wu, V. Apte, S. Bagchi, S. Garg, and N. Singh, "Intrusion Detection in Voice over IP Environments," International Journal of Information Security, Vol. 8, pp. 153–172, June 2009. Article (CrossRef Link).

[11] T. Dagiuklas, D. Geneiatakis, G. Kambourakis, D. Sisalem, S. Ehlert, J. Fiedler, J. Markl, M. Rokis, O. Botron, J. Rodriguez, and J. Liu, "General Reliability and Security Framework for VoIP Infrastructures," Tech. Rep. Deliverable D2.2, SNOCER COOP-005892, September 2005.

[12] H. Cha, J. Ryu, B. Roh, J. Kim, H. Jeong, "Detection of SIP De-Registration and Call-Disruption Attacks using a Retransmission Mechanism and a Countermeasure Scheme," IEEE SITIS'2008, Nov. 2008. Article (CrossRef Link).

[13] J. Ryu, B. Roh, M. Hong, H. Kim, J. Kim, "Analysis and Its Solution on Security Threats in SIP-based Mobility Support Environments," IEEE INOVATION'2008, Dec. 2008. Article (CrossRef Link).

[14] SIPp : SIP performance. http://sipp.sourceforge.net/.

[15] Asterisk : The open source telephony project. http://www.asterisk.org/.

[16] A. D. Keromytis, "A Comprehensive Survey of Voice over IP Security Research," IEEE Comm.

Surveys & Tutorials, accepted for publication.
[17] C. Holmberg, E. Burger, H. Kaplan, "Session Initiation Protocol (SIP) INFO and Package Framework," IETF RFC 6086, Jan. 2011.
[18] S. Donovan, "The SIP INFO Method," IETF RFC 2976, Oct. 2000.
[19] H. Schulzrinne, and E. Wedlund, "Application-layer mobility using SIP," ACM SIGMOBILE Mobile Computing and Communications Review, Vol.4, No. 3, pp.47-57, Jul. 2000. Article (CrossRef Link).
[20] C. Shen, and H. G. Schulzrinne, "On TCP-based SIP Server Overload," ACM IPTComm'2010, Aug. 2010. Article (CrossRef Link).
[21] J. Ryu, B. Roh B, and K. Ryu, "Detection of SIP Flooding Attacks based on the Upper Bound of the Possible Number of SIP Messages," KSII Tr. Internet and Information Systems, Vol.3, No.5, pp.423-574, Oct. 2009.

Jea-Tek Ryu received his B.S., M.S. and Ph.D degrees in Computer Engineering from Ajou University, Suwon, Korea, in 2005, 2007, and 2011, respectively. Since Feburary 2011, he has been with the Division of Information and Communication Technology at Korea Institute of Patent Information (KIPI), Seoul, Korea, as a researcher. His research interests include ubiquitous sensor networks, network security, multimedia network systems and data mining.
.

Byeong-hee Roh received a B.S. degree in Electronics Engineering from Hanyang University, Seoul, Korea, in 1987, and M.S. and Ph.D. degrees in Electrical Engineering from Korea Advanced Institute of Science and Technology (KAIST), Taejon, Korea, in 1989 and 1998, respectively. From 1989 to 1994, he was with Telecommunication Networks Laboratory, Korea Telecom, as a researcher. From February 1998 to March 2000, he worked with Samsung Electronics Co., Ltd., Korea, as a Senior Engineer. Since March 2000, he has been with the Graduate School of Information and Communication, Ajou University, Suwon, Korea, where he is currently an associate professor. During 2005, he was a visiting associate professor at Dept. of Computer Science, State University of New York, at Stony Brook, New York, USA. His research interests include mobile multimedia networking, network QoS, wireless sensor networks, network security, and military communications.

Ki-Yeol Ryu received B.S. degree in Computer Engineering from the Seoul National University in 1985 and M.S. and Ph. D. degree in Computer Engineering from KAIST in 1987 and 1992, respectively. He was a researcher at the Department of Computer Science in KAIST from 1992-1993. From 1993-1994, he was a visiting researcher at the Yonezawa Lab. at the Department of Information Science in the Tokyo University. He was also a visiting professor at the Department of Computer Science in the University of Colorado at Boulder for the year 2000. He is currently working at the Department of Information and Computer Engineering in Ajou University since 1994. His interests include application models and frameworks for large scale systems, ubiquitous computing, service-oriented computing, network security, etc.

Myungchul Yoon received the BS and MS degrees in electronics engineering from Seoul National University, Korea, in 1986 and 1988 respectively, and the Ph.D. degree in Electrical and Computer Engineering from The University of Texas at Austin in 1998. From 1988 to 2002, he was with Hynix Inc. Icheon, Korea as a technical research staff at Semiconductor R&D Lab. and Mobile Communication R&D Lab. From 2005 to 2006, he was with Daegu-Gyeongbuk Institute of Science and Technology (DGIST), Korea as a technical staff at the Information Technology R&D Division. Since 2006, he has been with the Department of Electronics Engineering, Dankook University, Cheonan, Korea, where he is a professor. His research interests are in mobile communication, wireless personal area networks (WPAN), embedded systems, and low-power VLSI design.