Implementation of Secure Hash Function over Session Description Protocol (SDP)

[1]Sheikh Mustaq Ahmmed,[2] Muhammad Minhazul Haque Bhuiyan, [3]AFM Sayem, [4]Md. Shihab Karim, [5]Md. Shakir Khan
[1]Lecturer,[2]Scenior Lecturer,[3] Engineer, [4] Associate Engineer, [5]Lecturer
[1]Depertment of Computer Science and Engineering,
[1]Leading University, Sylhet, Bangladesh.

***Abstract*— VoIP (Voice over Internet Protocol) is a highly popular way of internet telephony that exists in today's world. Most of the mobile service providers are moving towards IP telephony because of its high reliability, security and cheap establishment and maintenance cost. SIP (Session Initiation Protocol) is the protocol which is used to establish voice calls among two or multiple parties over the internet and is the most popular application layer protocol used for VoIP till date. An important part of SIP protocol remains another protocol named SDP (Session Description Protocol) that operates under SIP and is used to describe the multimedia session that is going to be established. However, this exchange of information is done in plaintext and can be exploited by hackers. In this work, we attempted to generate a hash to ensure data integrity which is part of the S/MIME standard. Then as a future goal we have set to encrypt the data and implement digital signature so that the sender is aware of the originator of the message.**

***Keywords*—VoIP, Session Initiation Protocol (SIP), Session Description Protocol (SDP), S/MIME, Real-time Transport Protocol, Secure Hash Algorithm**

I. INTRODUCTION

Session Initiation Protocol (SIP) [1] was created to establish services and provide support for the Internet Telephony. Many of these services, like addressing (telephone numbers) and busy-signals are familiar from the Public Switched Telephone Network (PSTN) but some are more advanced, like presence service and instant messaging. SIP is only a signaling protocol and cannot be used to carry actual data. However, SIP can act as a carrier for Session Description Protocol (SDP) [2] , which is used to negotiate parameters (IP addresses, ports, stream transport protocols, codec's etc) for the communicating parties.

The exploit that is the concern of this work is the information announcement in the SDP protocol that SIP uses. SDP is the format using which SIP announces the details of the session; e.g. who is the originator of the session, the description of the session, session ID, what media will be used in the session etc.

Secured media transport is an important idea in establishing VoIP communication as it aims to provide confidentiality, message authentication and integrity, and replay protection to the media (data) stream. The data stream typically carries voice datagram when it comes to VoIP. If the sent data is incomprehensible by any other apart from the recipients, it is called confidentiality. Message authentication means that the data received by the recipient was indeed intended for him/her from the expected sender. Data integrity implies that any modification of the data in transit will be detected by the recipient. An example of a secure media transport protocol used on VoIP communications is Secure Real-time Transport Protocol (SRTP), which is a profile of Real-time Transport Protocol (RTP).

SIP remains the most popular protocol to establish VoIP calls but it has certain security drawbacks. In our work, we aim to address one of its vulnerability and implement the solution up to a point that would ensure the integrity of the sender as the data travels over the network. SDP is part of the SIP protocol that shows certain vulnerable information about a session in plain-text which might be exploited by attackers to execute Man in the Middle attack and session hijacking. In this work, we aim to implement the sha256 algorithm into Minisip to ensure data integrity. The sha256 is a secured hash algorithm that generates a fixed length data corresponding to the original value. Then the original data and the hashed data would be integrated in a field and sent together. On the receiver end the same hash of the data will be produced and if both the hashes match then it is ensured that the data has not changed from sender to receiver.

The paper is organized as follows. In the next chapter we have given some brief discussion on background like, Session Initiation Protocol (SIP), Session Description Protocol (SDP), S/MIME and SHA-256. In the third section we pointed out the problem and set up a formal problem statement. We pointed out the several security threats and gave a brief discussion on those

aspects. This section includes the ways of pollution attack as well as the defense mechanisms. And then we have discussed about our experiment. We have given the details of our experiment, have discussed about our environment, used tools. We have discussed the used algorithms and explained them in this section. We have presented the results and outcomes of our experiment and evaluated them. The final section contains our conclusive decision and the reason for choosing this specific result.

II. BACKGROUND

To comprehend our work fully a few basic terms are needed to be explained. In the following sections a brief discussion over the key terms that we have used has been provided.

Voice over Internet Protocol (VoIP)

Voice over Internet Protocol is the assembling of voice into IP data. This data can then be transmitted over an IP network to an addressable (IP address) destination. VoIP systems employ session control protocols to control the set-up and tear-down of calls as well as audio codecs which encode speech allowing transmission over an IP network as digital audio via an audio stream. The codec used is varied between different implementations of VoIP (and often a range of codecs are used); some implementations rely on narrow band and compressed speech, while others support high fidelity stereo codecs.

The VoIP protocol stack is shown in figure 1. There are four layers: signaling, session description, key exchange and secure media (data) transport. This division is intentional as each layer is typically implemented by a separate protocol. Signaling is an application-layer (from the viewpoint of the underlying communication network) control mechanism used for creating, modifying and terminating VoIP sessions with one or more participants. Signaling protocols include Session Initiation Protocol (SIP) [1] ,H.323 and MGCP. Session description protocols such as SDP [2] are used for initiating multimedia and other sessions, and often include key exchange as a sub-protocol.

VoIP not only provides the same voice functions as traditional telephone systems, but operates over IP networks; it also adds new voice communications services. For example, "follow me" services enable a user to have his phone calls find him regardless of his location. With increasing VoIP deployment, the benefits are obvious. VOIP leverages existing data networks, saving the cost of building and operating a separate voice network. Voice traffic can often run on the data networks "for free", i.e. at zero incremental cost. It also saves significantly on long distance calls by routing VoIP calls through data networks. VoIP can provide more flexibility in operations and new services. It can also enhance productivity by allowing integrated voice.

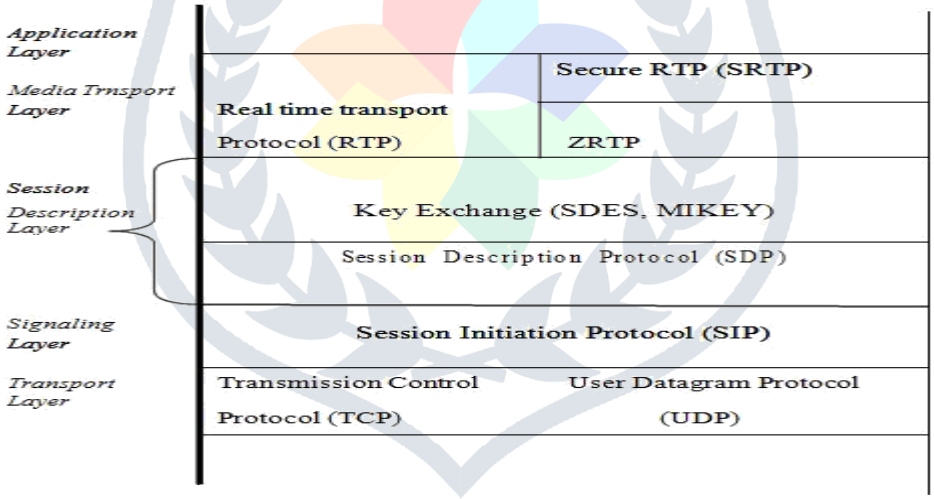

Figure 1: Voice-over-IP protocol stack

Session Initiation Protocol (SIP)

The SIP [1] is an application- layer control protocol that can establish, modify, and terminate multimedia sessions (conferences) such as Internet telephony calls or text based sessions. SIP can also invite participants to already existing sessions, such as multicast conferences. SIP works with both IPv4 and IPv6. SIP was originally designed by Henning Schulzrinne and Mark Handley starting in 1996. SIP appears as an alternative to ITU-T's (ITU Telecommunication Standardization Sector) H.323 [3] standard and proprietary standards such as the standard used by Skype. The details of the session, such as the type of media, codec, or sampling rate, are not described using SIP. Rather, the body of a SIP message contains a description of the session, encoded in some other protocol format. One such format is the Session Description Protocol (SDP) [2]. SIP helps to establishing and terminating multimedia Communication in the following ways:

User Location: finds the end system to be used for communication.

User availability: point out the willingness of the called party to engage in communications.

User capabilities: determination of the media and media parameters used.

Session setup: establishment of session parameters at both called and calling party, what is called "ringing".

Session management: including transfer and termination of session, modifying session parameters and invoking service parameters.

SIP Components

SIP has basically two components [4], SIP User Agent and SIP Network Servers. The user agent is the component in the end system and consists of two parts:

- The client element called user agent client(UAC) used for call initiation;
- The server element called the user agent server(UAS) that is used to answer calls.

The SIP servers' functions include resolving the name and providing user locations, as end users usually don't know the IP address or the hostname of the called party. Following are several examples of SIP servers:

Registrar server the registrar server receives Register requests from the users. The Register request associates the user's SIP address called a SIP URI (Uniform Resource Identifier) with the current machine where they are located. This association is stored by the Registrar in the Location Service (LS).

Proxy Server Users send their SIP requests to the Proxy Server, which forwards the requests to the next hop proxy server or to a proxy server close to the called user. The proxy server can also modify and add information in some parts of the SIP requests if required. A Domain Name System (DNS) Server can be used to find the location of the Proxy server.

Redirect Server The Redirect server receives the request from the clients, but unlike Proxy Servers, it does not forward the request to another server or the user. Rather, it sends back a response to the calling user with the information about the destination.

Making a Call

A standard call establishment procedure [4] can be described with the SIP trapezoid when UA1 wants to establish a Call to UA2, where the following SIP messages are involved:

- INVITE: used by the caller to initiate a call to the callee
- Trying: a response from the next hop server that the INVITE message is received and processed. Used by many UA but not mandatory.
- Ringing: alerting the caller that the callee has received the INVITE
- 200 OK: the request has succeeded e.g. call answered or hangs up
- ACK: establish the media session
- BYE: hang up call, this signal may be sent via the proxies. Figure 2. SIP Call establish parameters from the caller are encapsulated in the INVITE message in a SDP body, and the parameters chosen by the callee are encapsulated in the 200 OK messages. The message 12 ACK, 13 BYE, 14 200 OK, may go via the proxy. The DNS lookups that are needed in figure are not shown.

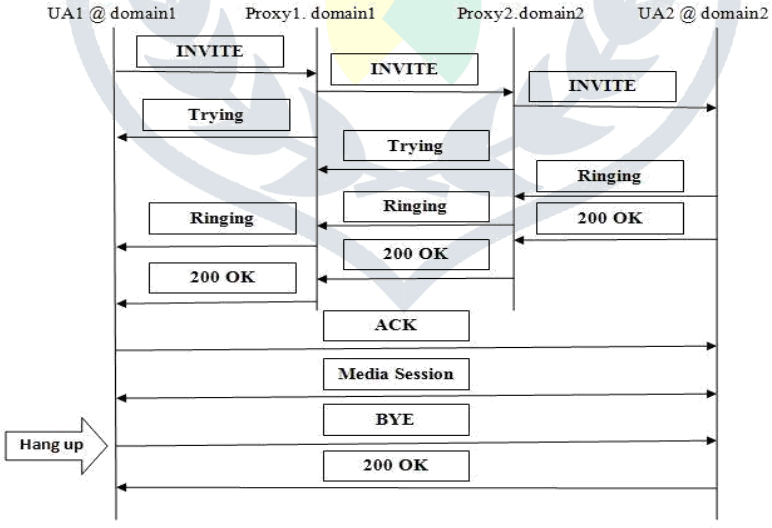

Figure 2: SIP Call Establish Preferred and supported session

```
Example of an INVITE message from caller John to callee Alice below

INVITE sip: alice@callee.org SIP/2.0
To: <sip:alice@callee.org>
From:<sip:john@doedomain.org>;tag=randomuniqu#
CallID: unique#@doedomain.org
```

```
Cseq: 4711 INVITE
Contact:<sip:john@johnspc.doedomain.org:5060;user=phone;transport=UDP>
MaxForwards: 70
Via: SIP/2.0/UDP johnspc.doedomain.org:5060;branch=z9hG4bKrandomunique#
Content Type: application/sdp
Content Length: 176
(Sdp body not shown)
```

The fields in the SIP body are as follows:

To: logical recipient of a request.
From: logical identity of the initiator of the request.
CallID: an unique identifier to group a series of messages. It must be the same for all requests and responses sent by either UA in a dialog.
Cseq: identification and order of transactions.
Contact: contains the URI at which the UA would like to receive requests.
MaxForwards: limits the number of hops a request can transit.
Via: indicates the transport used for the transaction and identifies the location where the Response is to be sent.
Content Type: indicates the media type of the message body sent to the recipient. Content Length: size of message body.

SIP Digest Authentication

SIP digest authentication mechanism [1] is a challenge-response paradigm which is used for client-to-client or client-to-proxy authentication like HTTP Digest mechanism [5]. In this scheme, the receiver can challenge the identity of the sender using a nonce value. A nonce is a unique string generated each time the "401 Unauthorized" message is sent and the realm specifies the digest algorithm used for the challenge. Sip digest authentication method uses five headers: www.Authenticate,Authorization,Proxy-authenticate,Proxy-authorization,Authorization-info.

If the origin server does not wish to accept the credentials sent with a request, it should return a 401 (Unauthorized) response. The response must include a WWW-Authenticate header field containing at least one (possibly new) challenge applicable to the requested resource. If a proxy does not accept the credentials sent with a request, it should return a 407 (Proxy authentication required). The response must include a Proxy-Authenticate header field containing a (possibly new) challenge applicable to the proxy for the requested resource after receiving this challenge, the client computes the response value using the nonce value, realm, username, shared secret with some optional parameters which is then included in the Authorization header in the new request message. By default, MD5 algorithm is used to compute this response value unless a different algorithm is specified in the realm parameter during the challenge. Figure 3 illustrates the SIP Digest authentication mechanism.

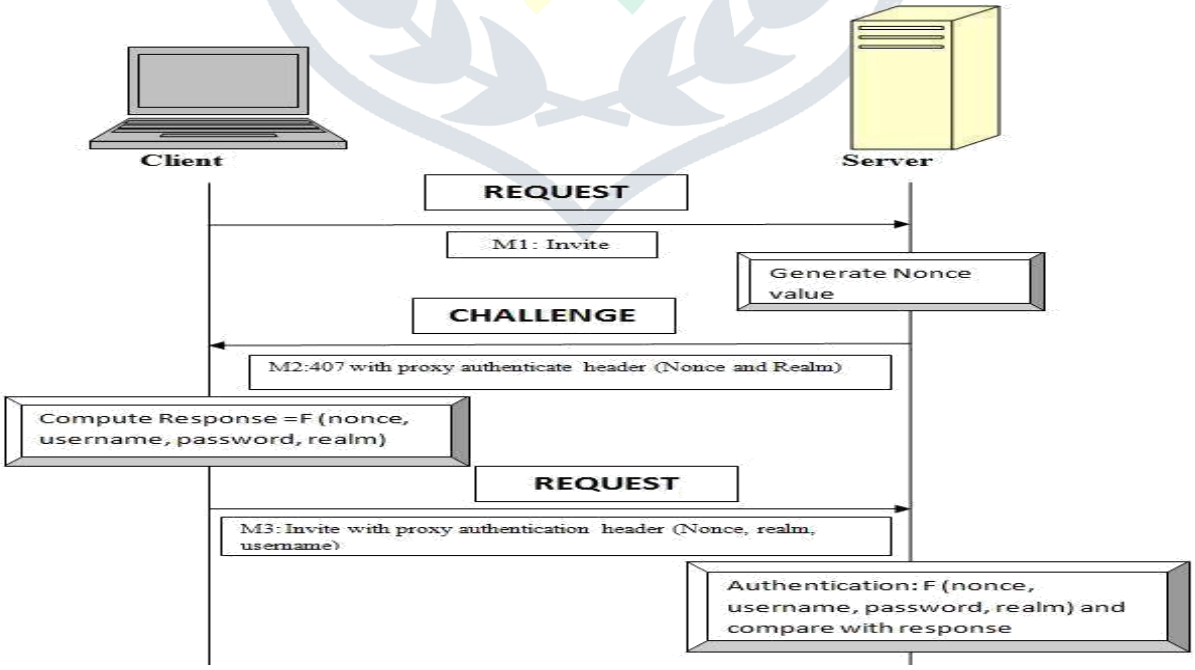

Figure 3: SIP Digest authentication mechanism

Session Description Protocol

SIP [2] is a format for describing multimedia session parameters for the purpose of session announcement, session invitation to allow the recipients of a session description to participate in the session. When initiating multimedia teleconferences, voice-over-IP calls, streaming video or other sessions, there is a requirement to convey media details, transport addresses, and other session description metadata to the participants. SDP provides a standard representation for such information, irrespective of how that information is transported. SDP is purely a format specification; it is independent of the transport layer and may be carried, for example, by SIP. SDP does not incorporate a transport protocol, and it is intended to use different transport protocols as appropriate, including the Session Initiation Protocol [1], Session Announcement Protocol [6], Real Time Streaming Protocol [7] , electronic mail using the MIME extensions, and the Hypertext.

SDP is primarily intended for use in an internetwork, although it is sufficiently general that it can describe conferences in other network environments. Media streams can be many-to-many. Sessions need not be continually active.

Several important pieces of information are mandatory within an SDP message:

- Session name
- Time(s) the session is active
- The media comprising the session
- The owner/originator of the session
- How to receive the media(address, ports, formats etc).

An SDP session description includes the following media information:

- The type of media (video, audio etc.)
- The transport protocol(RTP/UDP/IP,H3.320 etc.)
- The format of the media (H.261 video, MPEG video, etc)

Other optional information may be provided:

- Bandwidth to be used by the conference.
- The purpose of the session.
- Contact information for the person responsible for the session.
- Time zone information.
- Session attributes extending SDP.

SDP used by SIP

To establish a VoIP call the caller need to negotiate session parameters with the caller. A call can consist of several multimedia channels e.g. voice and video etc., where each channel needs a unique set of parameters that describes the session. This negotiation can be done with SIP. The caller sends a SDP body that is encapsulated in the SIP INVITE message, with proposed parameters. The callee responds with a SDP body in the OK message with the the chosen parameters, so in just one round trip a common pair is negotiated. How SDP is used by SIP is given bellow [8]:

```
INVITE sip: alice@callee.org
SIP/2.0 To: <sip:alice@callee.org>
From: <sip:john@doedomain.org>;
tag=randomunique# CallID:unique#@doedomain.org
Cseq: 4711 INVITE
Contact:<sip:john@johnspc.doedomain.org:5060;user=phone;transport=UDP>
MaxForwards:70
Via:SIP/2.0/UDPjohnspc.doedomain.org:5060;branch=z9hG4bKrandomunique#
ContentType:application/sdp
ContentLength:139
v=0
o= 123 123 IN IP4 johnspc.doedomain.org
s=Minisip session
c=IN IP4
johnspc.doedomain.org t=0 0
m=audio 32869 RTP/AVP 0
a= rtpmap:0 PCMU/8000/1
```

The fields in the SDP body are as follows:
v = (protocol version) but also marks the beginning of the session

```
description
o= (owner/creator and session identifier)
s= (session name)
c= (connection information not required if included in all media)
t= (time the session is active)
```

Hash Function

Has function takes a group of characters (called a key) and maps it to a value of a certain length (called a hash value or hash). The hash value is representative of the original string of characters, but its normally smaller than the original. Hash function is used for both encryption and decryption. In our experiment we use hash function (SHA 256) instead of public key cryptography because using a public key to sign the entire message is very inefficient if the message is very large. The solution is to let the sender sign a digest of the document instead of the whole document. The sender creates a miniature version or digests of the document and signs it, the receiver then checks the signature on the miniature. To create the digest of the message, we use hash function [12]. The hash function creates a fixed-size digest from a variable length message, as shown in figure 4. Has function must have two properties to guarantee its success. They are:

- Hashing is one way, the digest can only be created from the message, not vice versa.
- Hashing is one -to-one function; there is a little probability that two messages will create the same digest.

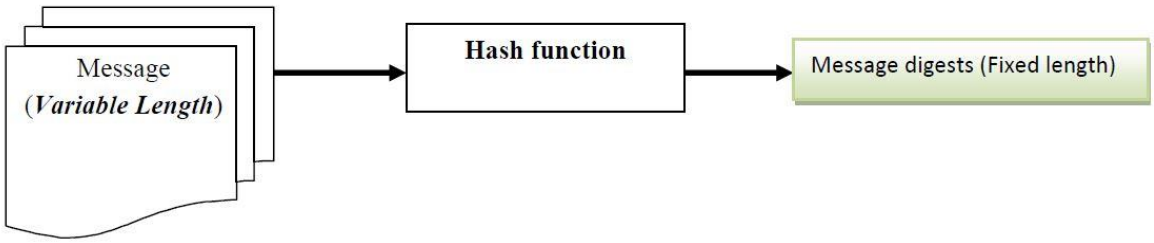

Figure 4: Hash Function

SHA-256 (Secure Hash Algorithm-256)

SHA-256 [13] is developed after SHA-1 in 2002 by NIST in order to match security levels offered by AFS. The message block size for SHA-256 is 512 bits and message digest size is 256 bits. Calculation of message digest is completed in 64 rounds. The general properties of SHA-256 are summarized in Table 1.

Table1: SHA-256 Summary

Message Size	$< 2_{64}$
Block Size	512 bits
Word Size	32 bits
Trans. Rounds	64
Mes. Digest	256 bits
Security	128 bits
#of chaining variables	8

SHA-256 calculation is completed in 64 rounds and 8 hash variables each of 32 bits are used. The word size of all the calculations is 32 bits. The padded message is processed by 512 bit blocks. This 512 bit block is composed of 16 message words. These 16 message words are expanded by means of functions and in each of the total 64 rounds a new message word is used. SHA-256 operates in the manner of MD4, MD5 and SHA-1. The message to be hashed first, and the steps are:

1. Padded with its length in such a way that the result is multiple of 512 bits long, and then
2. Parsed into 512-bit message blocks M[1] , M[2] …… ,M[N].

The message blocks are processed one at a time. Beginning with a fixed initial hash value H[0] , sequentially compute,

$$H(i) = H(i-1) + CM(i)(H(i1))$$

Where C is the SHA-256 compression function and + means word-wise mod 232 addition. H[N] is the hash of M.

S/MIME

Secure/Multipurpose Internet Mail Extensions (S/MIME)[14] is used for providing the end to end confidentiality and integrity of the MIME content to some extent by replicating the header fields in the MIME part . It is a standard for public key encryption and signing of MIME data. It provides security protocol for email that accomplishes two goals, namely: privacy and authentication. Privacy means that you can send an email message to someone and know that only the intended recipient can read

it. Authentication means that you can receive an email from someone and be certain that the message actually came from that sender. You can also combine these two techniques and send an email that is both private and authenticated; you know that no one else will read it, and the receiver knows that you really sent it.

S/MIME can be used by traditional mail user agents (MUAs) to add cryptographic security services to mail that is sent, and to interpret cryptographic security services in mail that is received. However, S/MIME is not restricted to mail; it can be used with any transport mechanism that transports MIME data, such as HTTP [5]. As such, S/MIME takes advantage of the object-based features of MIME and allows secure messages to be exchanged in mixed-transport systems.

Function of S/MIME

S/MIME provides the following cryptographic security services for electronic messaging applications: authentication, message integrity and non-repudiation of origin (using digital signatures) and privacy and data security (using encryption). S/MIME specifies the MIME type application/pkcs7-mime (S/MIME-type "enveloped-data") for data enveloping (encrypting) where the whole (prepared) MIME entity to be enveloped is encrypted and packed into an object which subsequently is inserted into an application/pkcs7-mime MIME entity. So we can say that S/MIME gives following thing:

- Secrecy – Only intended recipient can read the message. (A thick envelope and trustworthy couriers.)
- Authentication – Recipient knows the message came from the apparent sender. (An ink signature that you recognize.)
- Integrity – Recipient knows the message was not changed en route. (Un-erasable ink in a letter.)

Authentication of S/MIME

SIP message are capable of carrying the Multipurpose Internet Mail Extensions (MIME) [14] bodies. S/MIME is used for providing the end to end confidentiality and integrity of MIME content to some extent by replicating the header fields in the MIME part . This scheme can also be used for authentication services by signing the replicated header fields to verify the identity of the sender. The S/MIME can be used to protect MIME payloads, such as the Session Description Protocol (SDP) data which is embedded on the application/SDP MIME body, contained in the SIP message by using the types multipart/signed or application/pkcs7-mim. S/MIME can be also used to protect the whole package through the creation of an S/MIME tunnel. **Figure 5.** Illustrates the SIP S/MIME authentication mechanism.

In this process, SIP User Agent creates the SIP message and attaches a MIME body to it. Then the UA creates a multipart/signed S/MIME entity, that contains the MIME body and an application/pkcs7-signature S/MIME entity. Afterwards, UA signs the MIME body by using its private key and includes the public key certificate in the CMS object of the application/pkcs7-signature S/MIME entity and then it sends the message. The receiving SIP UA takes the message apart and tries to find the included public key certificate in its key ring. If a public key certificate, in the key ring, has a subject that matches the From1 header field in the SIP message, then the receiving SIP UA should compare it with the received certificate. If there is a discrepancy between them, the SIP UA should notify the user and acquire the user's permission before continuing the session.

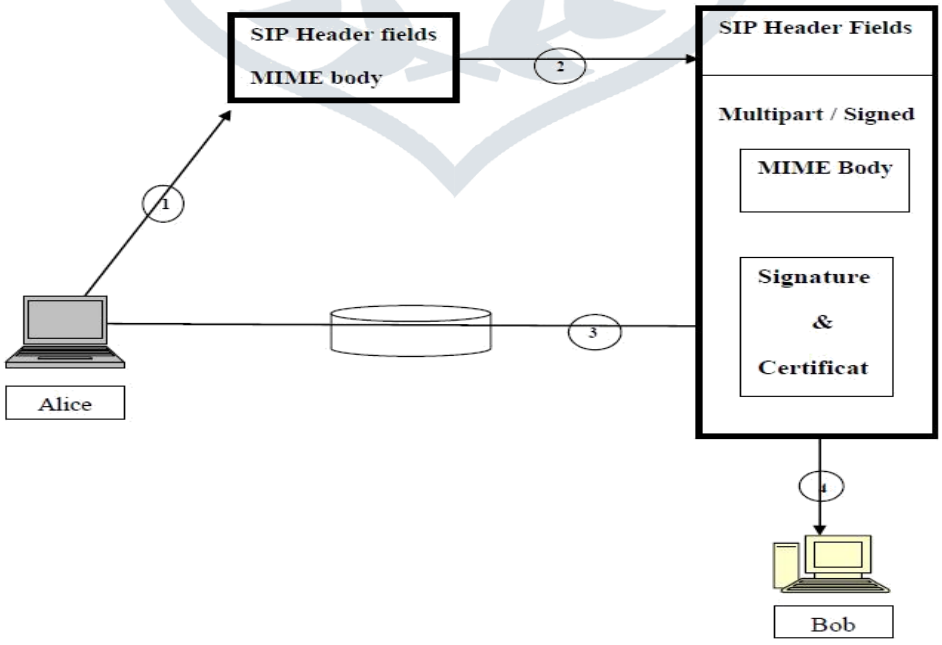

Figure 5: SIP S/MIME authentication process

MiniSIP

MiniSIP[11] is a SIP user agent which is developed by Erik Eliasson at KTH, Stockholm, Sweden. It is a SIP based soft phone, which works under LINUX. MiniSIP can work both on the iPAQ (tested on an HP iPAQ h5550 PDA) and on a workstation/laptop running Linux. It is designed based on Session Initiation protocol (SIP) and special security features are included in it. MiniSIP also supports multiple CODECs and this feature makes it very useful for both high and low quality connectivity. As it is open source software, a community from both universities and companies are working on it.

Minisip is divided into four independent subsystems. The graphical user interface (GUI) subsystem, policy subsystem, media subsystem and sip subsystem. The GUI subsystem is responsible for interacting with the user (initiating and answering calls). The policy subsystems is responsible for making decision about whether an incoming call should make the user phone alert or if it should be ignored. The media subsystem is responsible for sending and playing media streams during a call. The SIP signaling logic is implemented by sip subsystem. Details about these subsystems can be found in master thesis of Erik Eliasson[16].

The most important feature of Minisip is that it supports security issues in VoIP. Minisip provides the first public implementation of Secured Real Time Transport Protocol (SRTP) [7] and the first public implementation of Multimedia Internet Keying (MIKEY).

Let us have a closer look at the security features added by these. In MiniSIP, Mutual Authentication between the users is provided by the MIKEY support with the help of a key exchange. This key management is added to the INVITE message. The authentication of the user to the Registrar Server and by the Registrar to the user utilizes Transport Layer Security (TLS) [17]. One way this can be done is if the Registrar Server has a certificate installed on it and is presents this certificate to a user; the user verifies it through some Certificate Authority; once this is done, the user has to authenticate itself to the Registrar and this can be done with the help of a user name and password. This TLS support in MiniSIP is possible only if the Registrar Server also supports TLS. The same method can provide authentication of a user to a Proxy Server and the other way, again if the Proxy Server has TLS support. This has not been tested by use. Hop by hop authentication between the Proxy Servers is also possible through TLS.

Another security measure that is provided by MiniSIP is securing the SIP messages themselves as a user might not want this information to be disclosed to others. This feature is also achieved with TLS using Public Key Infrastructure (PKI) [18]. Encryption for the media information is provided in MiniSIP by SRTP. In this case, MIKEY provides the key for SRTP.

III. PROBLEM STATEMENT

Overview

The originator field in SDP is the main concern of this work as it gives out critical information about the person who wants to establish the communication.

```
o = <username> <session id> <version> <network type> <address type> <address>
```

The "o =" field gives the originator of the session (their username and the address of the user's host) plus a session id and session version. <username> is the user's login on the originating host, or it is "-" if the originating host does not support the concept of user ids. <username> must not contain spaces. <Session id> is a numeric string such that the tuple of <username>, <session id>, <network type>, <address type> and <address> form a globally unique identifier for the session.

An example of SDP description is:

v =0
o=mustaq 2890844526 2890842807 IN IP4 126.16.64.4
s= A secret message!
i= A session with your cashier
u= http://www..yourbank.com/youraccount
t=2873397496 2873404696
m= audio 49170 RTP/AVP

This information can be used against the sender and might prove to be critical against protection of vulnerable data. Two kinds of attacks can be formulated using the above data:

- **Registration Hijacking**: It occurs when an attacker impersonates a valid UA to a registrar and replaces the legitimate registration with its own address. This attack causes all incoming calls to be sent to the UA registered by the attacker.
- **RTP Data Redirecting**: RTP packet redirecting is the kind of an attack the SDP header is modified to redirect the caller to another destination. For example the packet is sniffed out using Scapy or any other sniffer that can be then used to modify and redirect the data to another user or may even be used to maintain the session by sending the BYE signal to the caller. The values are changed in SDP headers o(originator) field and them sent to the malicious user.

Problems to be addressed

- Ensure data integrity of SDP so that forgery of the session can be detected.
- Encrypt the session data so that the data is not understandable to the attacker.
- Sign the data digitally so that the receiver can be sure of the originator of the message.

IV. DESIGN AND IMPLEMENTATION

In this work, we tried to implement the sha256 algorithm into Minisip to ensure data integrity . The sha256 has function would take the data as a parameter and generate a fixed length hash for it. Then the original data and the hashed data would be integrated in a field and sent together.

The reason we opted for Minisp as client because Minisip is an open source soft phone client rich with security faciliteis. It also has a major advantage over other open source soft phones because of the massive security related development and research work it encompasses. Using Minisip as client and Asterisk as server we established pc to pc soft phone network and analyzed the data. The data exposed the vulnerability of SDP as SDP showed some user sensitive data in plaintext.

Environment Setup:

First up, Minisip was download and compiled in two computers that are residing in the same network. As the SIP proxy server, we used Asterisk (1.6.2.7). Asterisk is an open source SIP proxy which provides with PABX solutions, SIP telephony, Voicemail etc and also comes with a very intuitive CLI. As the packet tracer, we used Wireshark (1.2.11). All the software‟s except Minisip has been downloaded from Ubuntu 10.10‟s default repository.

Environment Setup:

Setting up Asterisk server consists of two steps. Two files are needed to be edited in order to use Asterisk as the SIP Proxy over a local network. They are:

- sip.conf (/etc/asterisk/sip.conf)
- extensions.conf(/etc/asterisk/extensions.conf)

There is an example given bellow:

```
sip.conf (/etc/asterisk/sip.conf)
[general]
port=5060
bindaddr=0.0.0.0
context=others
[2000]
type=friend
context=my-phones
secret=1234
host= dynamic
[2001]
type=friend
context=my-phones
secret=1234
host= dynamic
```

```
extensions.conf
(/etc/asterisk/extensions.conf)

[others]

[my-phones]
exten=>2000,1,Dial(SIP/2000)
exten=>2001,1,Dial(SIP/2001)
```

Design Overview:

The SDP header gets generated through a function call and the function returns a value of type string. The value then individually gets assigned to each field of sdp, but in an ordered fashion. Each sdp Header object has two properties, the type of value and priority. The type of value contains the type of an individual field and priority determines which object would come after which thus maintaining the order of the fields.

The design philosophy is very simplistic. We indentified the function that is responsible for generation the string data. Then while it was being generated, we passed its value to the hash function which produced the hash of the given data. After that we concatenated the value the data. A pictorial description is shown in the figure 6 :

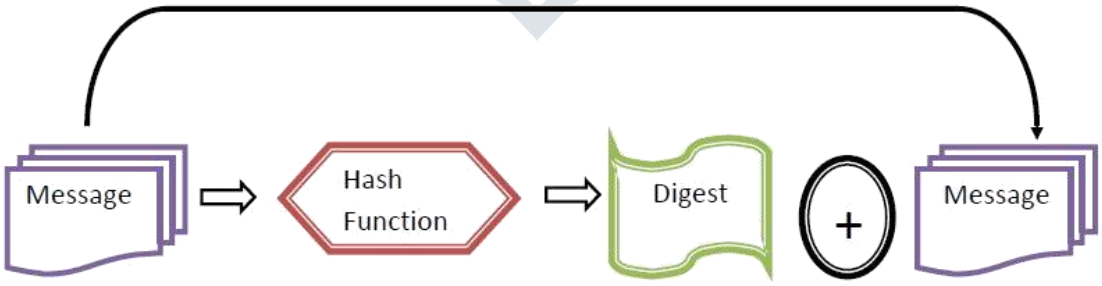

Figure 6: Proposed Solution

Experiment Algorithm:

The has function that we used was sha256. It was download from the author's (Olivier Gay) website and in the source file the implementation was named as “FIPS 180-2 SHA-224/256/384/512 implementation”. The implementation contained all the implementation of sha2 but to keep things simple and the code clean we kept only sha256 and its related functions inside the file. Figure 7 shows the sha256 hashing algorithm takes the value and passes it through the following functions to generate the hash.

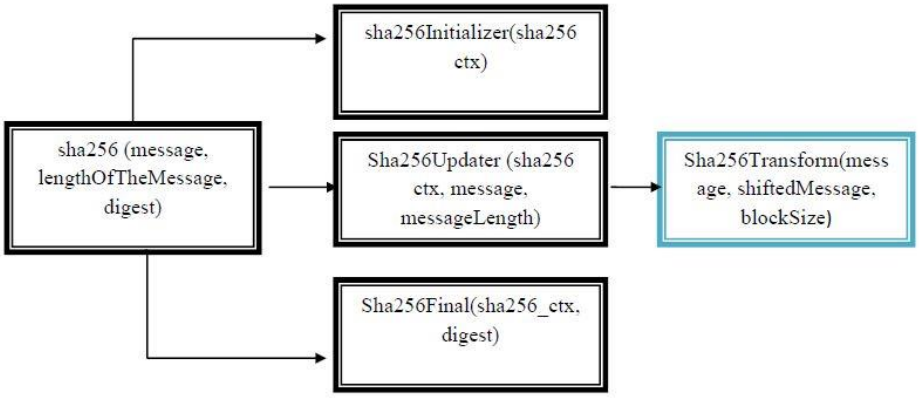

Figure 7: sha256 Function Calling sequence

Integration of Minisip and sha256:

To integrate MInisip and the sha256 implementation, we first declare the functions in the SdpHeader.h file where all the other declarations have been made. We also moved the custom type definitions and constants of sha256 to the SdpHeader.h.

Next, the sha256 main function was refactored to work with Minisip. For example, the main function was refactored to sha256Digest. The test vectors were removed and the function was modified to work with strings (prior to that, it could only work with constant character arrays). Necessary casting to character arrays and recasting to strings has been done so that the final digest outputs to string data. If there has been no casting then required string operations could not be done.

As a test value, we opted for the session id. The session_id is a string value that gets generated when the constructor to sdpHeader is called with a string type parameter. Then the generated value is passed on to the sdpHeader::getString() method to generate the SDP. The getString method is an abstract method declared in sdpHeader class so any subclass of sdpHeader (in this case, sdpHeaderO) is bound to implement the method.

We called the hash function when the session_id was being initialized. During initialization, we concatenated it with a separator (_) and then called the sha256Digest function with the session_id parameter to get the main value and the hash value together. The sha256Digest function has a return type of string so concatenation operations can be done on it. Although, the FIPS implementation uses constant character arrays; but it can be cast to change into string value.

V. RESULT AND DISCUSSION

First, we took two unmodified instances of Minisip and established a connection between them using Asterisk. Then we established a call from a pc to the other and caught and analyzed the packets that were exchanged during call establishment. For identifying changes in SDP header, we chose Wireshark as our packet analyzer. Wireshark identified the SDP packets and showed the values in the message body.

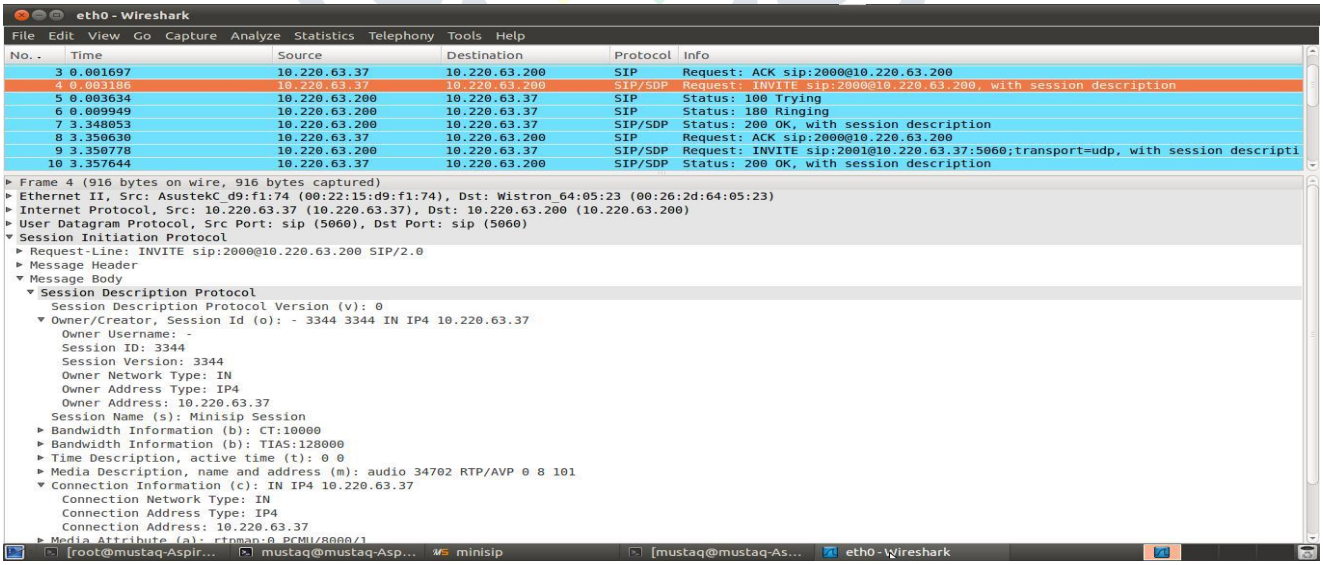

Figure 8: SDP Exposing Vulnerable Data

From the figure 8 we can see that the SDP is giving out session sensitive information in plain text. There exists no mechanism to identify if the packet has changed during call establishment.

With our work, we tried to achieve that as we changed the way Minisip sends the data by attaching the hash of the

intended message. After necessary modification to Minisip, we established the connection again between the two computers and the generated SDP with its content is shown in figure 9.

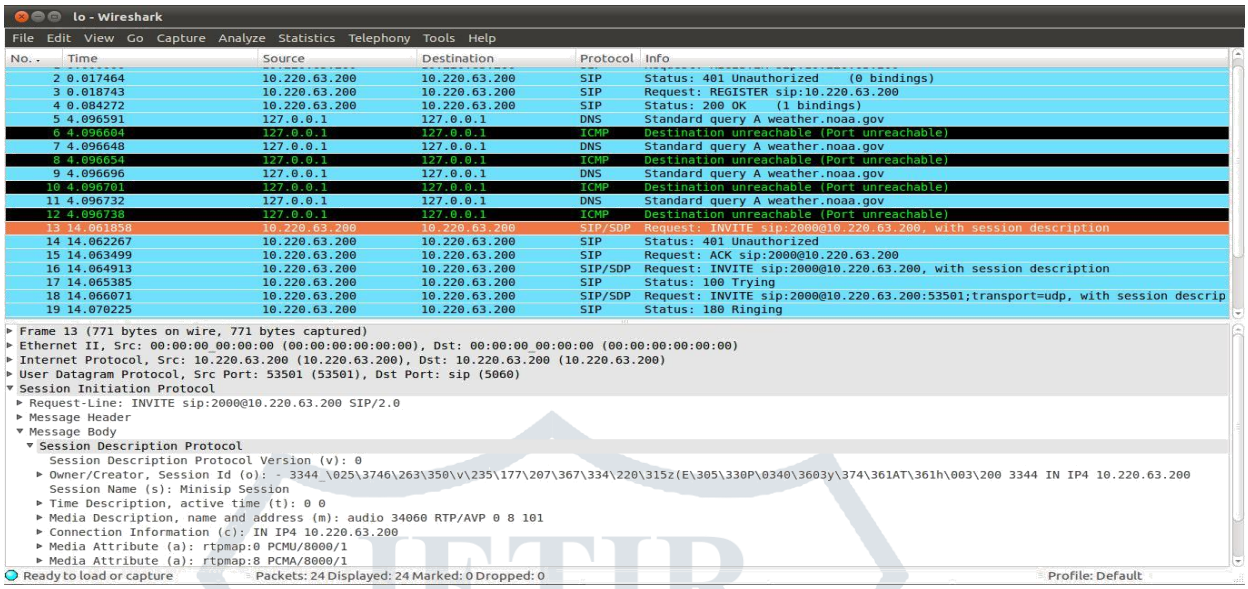

Figure 9: Session Id with appended hash

From the previous figure, if we take a look at the session_id value in the originator field, we can see that the value has been appended with a fixed length of string. This is the hash of the function. Now if the original message gets changed during exchange of SDP then it is possible to identify in the receiving end by taking the value and generating a hash of it and comparing the value that is attached to the real value.

VI. Conclusion and Future Plan

VoIP has gained a huge attention worldwide because of its cheap establishment, cost effective maintenance, control and flexibility and massive portability. But as it is gaining popularity, so is increasing the risks associated with it as attackers will target it for their personal benefits. So there is a growing concern about the security and vulnerability of such mechanism and more researches are being dedicated towards it as time passes.

SIP remains the most popular protocol to establish VoIP calls but it has certain security drawbacks. In our work, we tried to address one of its vulnerability and implemented the solution up to a point that would ensure the integrity of the sender as the data travels over the network. SDP is part of the SIP protocol that shows certain vulnerable information about a session in plain-text which might be exploited by attackers to execute Man in the Middle attack. An extensive analysis has been performed in order to detect the location and origination of SDP headers and the options has been weighed on to which algorithm is to be used in order to obtain data integrity.

The Minisip user agent has been modified and integrated with sha256 algorithm, a secure hash function that generates a fixed length string in reference to the corresponding value. This ensures that any change in the original value can be detected to the other end. This is also part of the S/MIME implementation standard that we believe is attainable and thus it will help attain a completely secured and standardized form of secured SDP.

Future Plan:

The current work can be used as a platform to make the SDP a complete secured protocol. By using public key based encryption we can ensure that the data stays hidden from the attacker. Encryption is also needed so that the attacker cannot change the hash value and even if they do, that they have no control over the process. This also ensures that RTP redirection attacks are harder to perform and require user specific research before conducting an attack.

Digital signature is also very vital in order to ensure authenticity of the sender. In digital signature, the receiver of the message can decrypt it using the sender's public key. The key generation has to be done by a trusted certificating authority.

The impact of efficiency and latency in call establishment due to cryptographic calculations also needs to be answered in future.

REFERENCES

[1] J.Rosenberg, H. Schulzrinne, G Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley and E. Schooler, "SIP: Session Initiation Protocol," RFC 3261,June 1999 [online.] Available: http://www.faqs.org/rfcs/rfc3261.html
[2] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998
[3] "ITU-T Recommendation H.323: Packet-based Multimedia Communications Systems," International Telecommunications Union, Jun. 2006.
[4] G. Q. Maguire Jr., „2G1325/2G5564 Practical Voice Over IP (VoIP): SIP and related protocols, Spring 2004, Period 3", Lecture notes. <http://www.imit.kth.se/courses/2G1325/VoIP-2004.pdf> (May 20, 2004)
[5] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A. and L. Stewart, "HTTP authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.[online].Available: http://www.faqs.org/rfcs/rfc2617.html
[6] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.
[7] Schulzrinne, H., Rao, A., and R. Lanphier , "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.
[8] J. Rosenber, "An Offer/Answer Model with the Session Description Protocol (SDP) " June 2002. [online].Available: http://www.ietf.org/rfc/rfc3264.txt
[9] Anoop Ms, "Public Key Cryptography" Application Algorithm and Mathematical Procedure. [online].Available:http://www.tataelxsi.com/whitepapers/pub_key2.pdf?pdf_id=public_key_TEL.pdf
[10] RSA Laboratories, http://www.rsa.com/RSALABS/node.asp?id=2259 Last visited: September 2011. 33
[11] KEMAL BICAKI " On the efficiency of authentication protocols, digital signature and their applications: A top-down approach" January 2003 [online.] Available : http://etd.lib.metu.edu.tr/upload/1101500/index.pdf
[12] M Bellare " Keying Hash Function for Message Authentication" June 1996 [online.] Available http://cseweb.ucsd.edu/~mihir/papers/kmd5.pdf
[13] Description of SHA-256, SHA-384, SHA-512. Available: http://www.iwar.org.uk/comsec/resources/cipher/sha256-384-512.pdf
[14] Abdullah Al Hasib, Abdullah Azfar and Md. Sarwar Morshed "Towards Public Key Infrastructure less authentication in Session Initiation Protocol" Jan 2010. Available: www.ijcsi.org/papers/7-1-2-10-17.pdf
[15] Minisip Website : http://www.minisip.org/
[16] Erik Eliasson, "Secure Internet Telephony: Design, Implementation, and Performance Measurement", Licentiate thesis, Royal Institute of Technology (KTH), School of Information and Communication Technology, June 2006.
[17] E. Rescorla, N. Modadugu, "Datagram Transport Layer Security," RFC 4347 April 2006 [Online]. Available: http://www.faqs.org/rfcs/rfc4347.html
[18] Johan Bilien, „Key Agreement for Secure Voice over IP", Master of Science Thesis At IMIT/KTH,December 2003. <ftp://ftp.it.kth.se/Reports/DEGREE-PROJEC REPORTS /031215-Johan-Bilien- report-final-with-cover.pdf > (May 31, 2004).