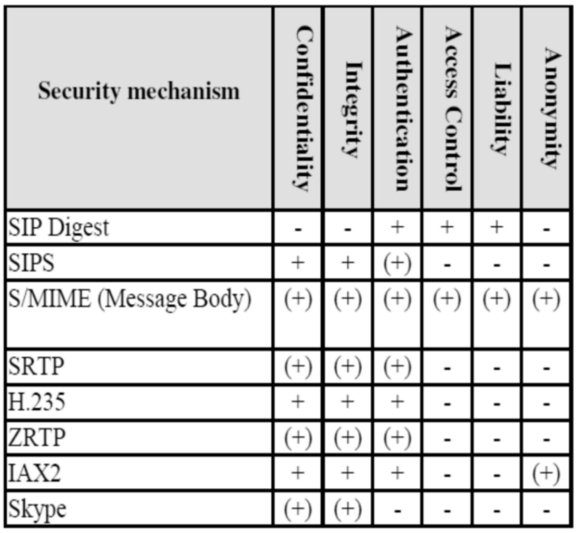


| Security mechanism | Confidentiality | Integrity | Authentication | Access Control | Liability | Anonymity |
|---|---|---|---|---|---|---|
| SIP Digest | - | - | + | + | + | - |
| SIPS | + | + | (+) | - | - | - |
| S/MIME (Message Body) | (+) | (+) | (+) | (+) | (+) | (+) |
| SRTP | (+) | (+) | (+) | - | - | - |
| H.235 | + | + | + | - | - | - |
| ZRTP | (+) | (+) | (+) | - | - | - |
| IAX2 | + | + | + | - | - | (+) |
| Skype | (+) | (+) | - | - | - | - |

Table (3): Overview of VoIP Security Protocol

## IV. PROBLEMS IN SIP AUTHENTICATION

We have two major weaknesses in HTTP digest authentication in SIP. The first missing security issue is the lack of securing all headers and parameters in SIP which would possibly need protection. The second security weakness, relating to digest authentication, is the requirement of pre-existing user configuration on servers, which does not scale well [9].Though, for authenticating in cellular mobile communication, it provides simple authentication. This solution enables a mutual authentication between any devices and the network. This security policy requires a shared secret key and a shared cryptographic algorithm that exist in SIP. So, pre-share keys are one of the main problems for security distributed keys and cause algorithm load for encode and decode security information packet. S/MIME in SIP is used on carrying signed or encrypted replication of headers and authenticating users. This mechanism lacks the public key distribution problem, which means that the public keys used in authentication are difficult to distribute and maintain. The public key infrastructure is also susceptible to man-in-the middle attack [3]. It uses several hash computations and server certificates to ensure security. This causes overhead and reduction in performance.

1) One of the crucial security issues faced by current SIP Protocol in how to authenticate end user identities. In SIP , the identity of an end user is defined by its SIP Uniform Identifier(URI), Which typically has a canonical address-of record(AoR), *From* field, for ex: sip:alice@10.0.2.1 . There are several places within a SIP request where an end user can express his identity. For example, the user populated *From* header field in the SIP INVITE message. Hence an end user can spoof his identity by inserting a false address in the *From* field and there is no mechanism for verifying that field.
2) Source IP cannot be determine because VoIP assign IP address dynamically. Since request and response each of the header contains the field name as “Via:” which store the path of each proxy so that receiver can communicate to the same path but the problem is that what about the path from UAC to proxy, proxy will not store that address, only SIP address of the caller (UAC) will be in the header but not any other information. So that makes it difficult to reach to the caller. In the proposed SIP stack they have the proxy which has the authentication certificate. but they do not have the client which has its own certificate which uniquely identifies the user and its public key and private key. Another problem is the path between caller (UAC) to proxy is still unsecure so that the authentication is also unsecure because man in the middle attack can be possible between UAC and proxy. If the user is authorized then it won’t create any problem but if the user is attacker and he wants to attack by doing spoofed call then we can never determine that who has done attack and from where it has been initiated because VoIP uses dynamic IP address.
3) In the existing mechanism, SIP can provide client-to-client protection only at the time of media exchange using RTP and not at the time of authentication. So to lessen the impact of the vulnerabilities from all the above mentioned issues. There is a need for a stronger authentication mechanism.

## V. PROPOSED APPROACH

### A. *Proposed Authentication Mechanism*

The proposed mechanism has the following features, as shown below:

1) *Proposes a transitive authentication mechanism*
    1) A new SIP header
    2) An authentication service running on proxies and UAC both
    3) Only relevant for SIP requests, not responses

### B. *Steps in header verification*

1) Acquire certificate for domain either stored or retrieved
2) Validate certificate, determine signer's authority over “From”
3) Verify signature
4) Validate Date, Contact, Call-ID

We will start explaining the concept starting from the problem mentioned below:

Suppose if Alice sends an INVITE message along with her signature to Bob, then Bob would require Alice’s public key to verify her signature. Hence in this case, Bob is faced with two problems:

1) How and from where would Bob retrieve Alice’s public key?
2) And how can Alice be sure that the key is actually Bob’s public key and not the attacker’s public key?

Public Key Infrastructure (PKI) helps solve the above problem. The purpose of PKI is to help Bob retrieve Alice’s public key, and to assure Alice that the key really belongs to Alice and not of somebody else. PKI distributes public keys using public key certificates.

However, with VoIP communications the audio signal is converted in several encrypted digital 'packets' which are sent separately via different routes across the internet, only re-collating when they reach the other user's computer. This means that there is no exchange through which all the information passes and so traditional methods of interception are ineffective. Instead, the problem of interception becomes one for computer forensic analysts.

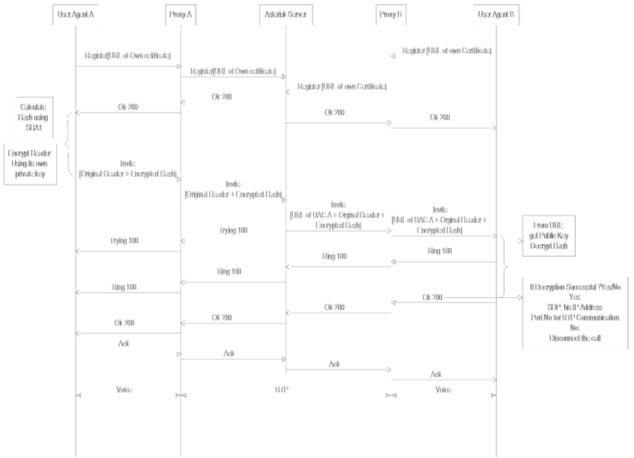
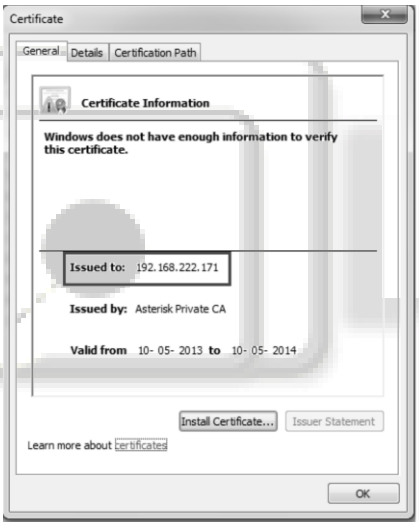

User Agent A
Proxy A
Asterisk Server
Proxy B
User Agent B
Register[URL of Own certificate]
Register[URL of Own certificate]
Register [URL of own Certificate]
Register [URL of own Certificate]
Ok 200
Ok 200
Ok 200
Ok 200
Calculate Hash using SHA1
Encrypt Header Using its own private key
Invite [Original Header + Encrypted Hash]
Invite [Original Header + Encrypted Hash]
Invite [URL of UAC A + Original Header + Encrypted Hash]
Invite [URL of UAC A + Original Header + Encrypted Hash]
From URL: get Public Key Decrypt Hash
Trying 100
Trying 100
Ring 180
Ring 180
Ring 180
Ring 180
If Decryption Successful ?Yes/No
Yes:
SDP: his IP Address
Port No for RTP Communication
No:
Disconnect the call
Ok 200
Ok 200
Ok 200
Ok 200
Ack
Ack
Ack
Ack
Voice
RTP
Voice

Fig (1): Proposed Approach

## VI. PERFORMANCE EVALUATION, RESULT & ANALYSIS

### A. Analysis of Modified SIP Protocol

The end user certificate and a server certificate has been created by running the “riddhi_certificate_create” script in the contrib/scripts shown as below.



Certificate

Certificate Information

Windows does not have enough information to verify this certificate.

Issued to: 192.168.222.171

Issued by: Asterisk Private CA

Valid from 10- 05- 2013 to 10- 05- 2014

Learn more about certificates

Fig (2)

In the development, standardization and implementation of LTE Networks based on Orthogonal Freq. Division Multiple Access (OFDMA), simulations are necessary to test as well as optimize algorithms and procedures before real time establishment. This can be done by both Physical Layer (Link-Level) and Network (System-Level) context. This paper proposes Network Simulator 3 (NS-3) which is capable of evaluating the performance of the Downlink Shared Channel of LTE networks and comparing it with available MATLAB based LTE System Level Simulator performance.

KEY WORDS--3GPP, LTE, Downlink, NS-3, Simulator, MAC, PHY

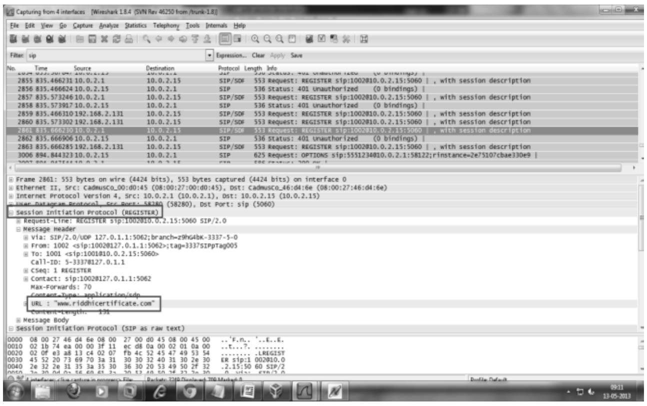
## VII. INTRODUCTION

The Long Term Evolution (LTE) standard specified by the 3rd Generation Partnership Project (3GPP) is a new mobile communication technology, which is evolution of the Universal Mobile Telecommunications System (UMTS) and High-Speed Packet Access (HSPA) systems. LTE intends to deliver high speed data and multimedia services to next generation. LTE is also backward compatible with the CDMA family of technologies and thereby enables even CDMA operators to move to this technology. The main reasons for these changes in the Radio Access Network (RAN) system design are the need to provide higher spectral efficiency, lower delay, and more multi-user flexibility than the currently deployed networks. LTE supports scalable carrier bandwidths, from 1.4 MHz to 20 MHz and supports both frequency division duplexing (FDD) and time-division duplexing (TDD). The IP-based network architecture called the Evolved Packet Core (EPC) is designed to replace the GPRS Core Network. The LTE device has been conceived as a container of several entities: the IP classifier, the RRC entity, the MAC entity and the PHY layer. The core of the LTE module is composed by both MAC and PHY layers of an LTE device.

The Evolved Packet Core comprises the Mobility Management Entity (MME), the Serving Gateway (SGW), and the Packet Data Network Gateway (PGW). The MME is responsible for user mobility, intra-LTE handover, and tracking and paging procedures of User Equipment (UEs) upon connection establishment. The main purpose of the SGW is, instead, to route and forward user data packets among LTE nodes, and to manage handover among LTE and other 3GPP technologies. The PGW interconnects LTE network with the rest of the world, providing connectivity among UEs and external packet data networks. The LTE access network can host only two kinds of node: the UE (that is the end-user) and the eNB. Note that eNB nodes are directly connected to each other (this speeds up signaling procedures) and to the MME gateway. The eNB is the only device in charge of performing both radio resource management and control procedures on the radio interface. Figure 1 shows Service Architecture Evolution in LTE network [7].

After certificate generation, now we will try to establish a call between two of the clients using the certificates and then analyze the call for the resultant scenario. For that we have to recompile our modified SIP code and run SIP phone again.

Once a call has been established will capture the VoIP Packets using WIRESHARK and analyze the REGISTER packet. it will show that user is able to add his URL.



Capturing from 4 interfaces [Wireshark 1.8.4 (SVN Rev 46250 from /trunk-1.8)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save

Filter: sip

```
No.  Time       Source          Destination  Protocol Length Info
2855 835.466231 10.0.2.1        10.0.2.15    SIP/SDP  553 Request: REGISTER sip:1002@10.0.2.15:5060 | , with session description
2856 835.466624 10.0.2.15       10.0.2.1     SIP      536 Status: 401 Unauthorized    (0 bindings) |
2857 835.573246 10.0.2.1        10.0.2.15    SIP/SDP  553 Request: REGISTER sip:1002@10.0.2.15:5060 | , with session description
2858 835.573917 10.0.2.15       10.0.2.1     SIP      536 Status: 401 Unauthorized    (0 bindings) |
2859 835.466310 192.168.2.131   10.0.2.15    SIP/SDP  553 Request: REGISTER sip:1002@10.0.2.15:5060 | , with session description
2860 835.573302 192.168.2.131   10.0.2.15    SIP/SDP  553 Request: REGISTER sip:1002@10.0.2.15:5060 | , with session description
2861 835.666230 10.0.2.1        10.0.2.15    SIP/SDP  553 Request: REGISTER sip:1002@10.0.2.15:5060 | , with session description
2862 835.666906 10.0.2.15       10.0.2.1     SIP      536 Status: 401 Unauthorized    (0 bindings) |
2863 835.666285 192.168.2.131   10.0.2.15    SIP/SDP  553 Request: REGISTER sip:1002@10.0.2.15:5060 | , with session description
3006 894.844323 10.0.2.15       10.0.2.1     SIP      625 Request: OPTIONS sip:5551234@10.0.2.1:58122;rinstance=2e75107cbae330e9 |

⊞ Frame 2861: 553 bytes on wire (4424 bits), 553 bytes captured (4424 bits) on interface 0
⊞ Ethernet II, Src: CadmusCo_00:d0:45 (08:00:27:00:d0:45), Dst: CadmusCo_46:d4:6e (08:00:27:46:d4:6e)
⊞ Internet Protocol Version 4, Src: 10.0.2.1 (10.0.2.1), Dst: 10.0.2.15 (10.0.2.15)
⊞ User Datagram Protocol, Src Port: 58280 (58280), Dst Port: sip (5060)
⊟ Session Initiation Protocol (REGISTER)
  ⊞ Request-Line: REGISTER sip:1002@10.0.2.15:5060 SIP/2.0
  ⊟ Message Header
    ⊞ Via: SIP/2.0/UDP 127.0.1.1:5062;branch=z9hG4bK-3337-5-0
    ⊞ From: 1002 <sip:1002@127.0.1.1:5062>;tag=3337SIPpTag005
    ⊞ To: 1001 <sip:1001@10.0.2.15:5060>
      Call-ID: 5-3337@127.0.1.1
    ⊞ CSeq: 1 REGISTER
    ⊞ Contact: sip:1002@127.0.1.1:5062
      Max-Forwards: 70
      Content-Type: application/sdp
    ⊞ URL : "www.riddhicertificate.com"
      Content-Length:   131
  ⊞ Message Body
⊟ Session Initiation Protocol (SIP as raw text)

0000  08 00 27 46 d4 6e 08 00  27 00 d0 45 08 00 45 00   ..'F.n.. '..E..E.
0010  02 1b 74 ea 00 00 3f 11  ec d8 0a 00 02 01 0a 00   ..t...?. ........
0020  02 0f e3 a8 13 c4 02 07  fb 4c 52 45 47 49 53 54   ........ .LREGIST
0030  45 52 20 73 69 70 3a 31  30 30 32 40 31 30 2e 30   ER sip:1 002@10.0
0040  2e 32 2e 31 35 3a 35 30  36 30 20 53 49 50 2f 32   .2.15:50 60 SIP/2
```

Profile: Default

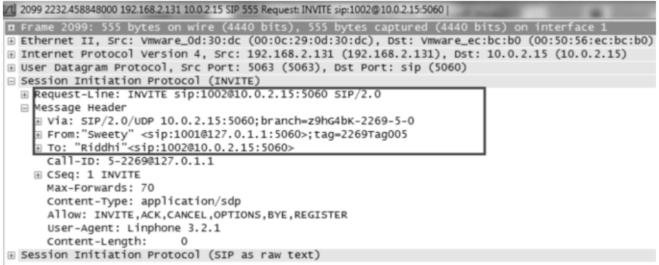
09:11
13-05-2013

Fig (3): modified SIP with URL

### *A. Modified SIP REGISTER Packet*

The following figure shows the packet captured during a normal VoIP call scenario.

#### *1) Client (UAC A) to server normal*

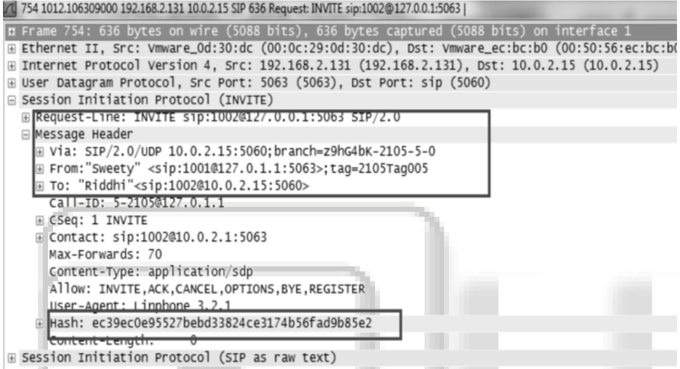


```
2099 2232.458848000 192.168.2.131 10.0.2.15 SIP 555 Request: INVITE sip:1002@10.0.2.15:5060 |
⊞ Frame 2099: 555 bytes on wire (4440 bits), 555 bytes captured (4440 bits) on interface 1
⊞ Ethernet II, Src: Vmware_0d:30:dc (00:0c:29:0d:30:dc), Dst: Vmware_ec:bc:b0 (00:50:56:ec:bc:b0)
⊞ Internet Protocol Version 4, Src: 192.168.2.131 (192.168.2.131), Dst: 10.0.2.15 (10.0.2.15)
⊞ User Datagram Protocol, Src Port: 5063 (5063), Dst Port: sip (5060)
⊟ Session Initiation Protocol (INVITE)
  ⊞ Request-Line: INVITE sip:1002@10.0.2.15:5060 SIP/2.0
  ⊟ Message Header
    ⊞ Via: SIP/2.0/UDP 10.0.2.15:5060;branch=z9hG4bK-2269-5-0
    ⊞ From:"Sweety" <sip:1001@127.0.1.1:5060>;tag=2269Tag005
    ⊞ To: "Riddhi"<sip:1002@10.0.2.15:5060>
      Call-ID: 5-2269@127.0.1.1
    ⊞ CSeq: 1 INVITE
      Max-Forwards: 70
      Content-Type: application/sdp
      Allow: INVITE,ACK,CANCEL,OPTIONS,BYE,REGISTER
      User-Agent: Linphone 3.2.1
      Content-Length:     0
⊞ Session Initiation Protocol (SIP as raw text)
```

Fig. (4) SIP INVITE from Client (UAC A) to Server

After modifying the code, when client will establish a call, at that time after capturing the packet (client to server) and analyzing it, it shows an extra field showing the hash value.

#### *2) Modified SIP INVITE packet Client (UAC A) to Server*

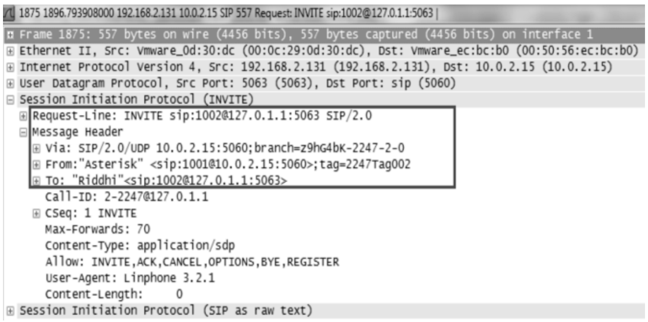


```
754 1012.106309000 192.168.2.131 10.0.2.15 SIP 636 Request: INVITE sip:1002@127.0.0.1:5063 |
⊞ Frame 754: 636 bytes on wire (5088 bits), 636 bytes captured (5088 bits) on interface 1
⊞ Ethernet II, Src: Vmware_0d:30:dc (00:0c:29:0d:30:dc), Dst: Vmware_ec:bc:b0 (00:50:56:ec:bc:b0
⊞ Internet Protocol Version 4, Src: 192.168.2.131 (192.168.2.131), Dst: 10.0.2.15 (10.0.2.15)
⊞ User Datagram Protocol, Src Port: 5063 (5063), Dst Port: sip (5060)
⊟ Session Initiation Protocol (INVITE)
  ⊞ Request-Line: INVITE sip:1002@127.0.0.1:5063 SIP/2.0
  ⊟ Message Header
    ⊞ Via: SIP/2.0/UDP 10.0.2.15:5060;branch=z9hG4bK-2105-5-0
    ⊞ From:"Sweety" <sip:1001@127.0.1.1:5063>;tag=2105Tag005
    ⊞ To: "Riddhi"<sip:1002@10.0.2.15:5060>
      Call-ID: 5-2105@127.0.1.1
    ⊞ CSeq: 1 INVITE
    ⊞ Contact: sip:1002@10.0.2.1:5063
      Max-Forwards: 70
      Content-Type: application/sdp
      Allow: INVITE,ACK,CANCEL,OPTIONS,BYE,REGISTER
      User-Agent: Linphone 3.2.1
    ⊞ Hash: ec39ec0e95527bebd33824ce3174b56fad9b85e2
      Content-Length:     0
⊞ Session Initiation Protocol (SIP as raw text)
```

Fig. (5) Modified SIP INVITE from Client to Server

The below figure shows the comparative analysis of the hash generated manually for the header packet to the hash generated by client in SIP packet and both the values show same result. This again shows the normal call establishment between Servers to client (UAC B). It has been added to show a comparative analysis of the packets virtue to the header fields modified and the proposed result

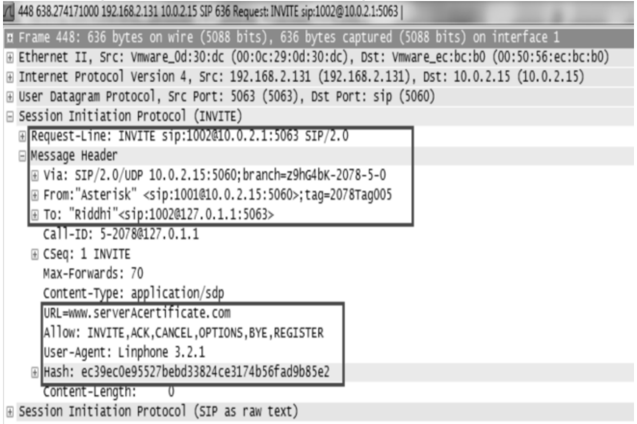
When client will establish a call, at that time after capturing the packet (server to client) and analyzing it, it shows URL field which is added by server and also shows the hash already generated by client.



```
1875 1896.793908000 192.168.2.131 10.0.2.15 SIP 557 Request: INVITE sip:1002@127.0.1.1:5063 |
⊞ Frame 1875: 557 bytes on wire (4456 bits), 557 bytes captured (4456 bits) on interface 1
⊞ Ethernet II, Src: Vmware_0d:30:dc (00:0c:29:0d:30:dc), Dst: Vmware_ec:bc:b0 (00:50:56:ec:bc:b0)
⊞ Internet Protocol Version 4, Src: 192.168.2.131 (192.168.2.131), Dst: 10.0.2.15 (10.0.2.15)
⊞ User Datagram Protocol, Src Port: 5063 (5063), Dst Port: sip (5060)
⊟ Session Initiation Protocol (INVITE)
  ⊞ Request-Line: INVITE sip:1002@127.0.1.1:5063 SIP/2.0
  ⊟ Message Header
    ⊞ Via: SIP/2.0/UDP 10.0.2.15:5060;branch=z9hG4bK-2247-2-0
    ⊞ From:"Asterisk" <sip:1001@10.0.2.15:5060>;tag=2247Tag002
    ⊞ To: "Riddhi"<sip:1002@127.0.1.1:5063>
      Call-ID: 2-2247@127.0.1.1
    ⊞ CSeq: 1 INVITE
      Max-Forwards: 70
      Content-Type: application/sdp
      Allow: INVITE,ACK,CANCEL,OPTIONS,BYE,REGISTER
      User-Agent: Linphone 3.2.1
      Content-Length:     0
⊞ Session Initiation Protocol (SIP as raw text)
```

Fig. (5) SIP INVITE from Server to Client (end user)

#### *3) Modified SIP INVITE Packet from Server to Client*



```
448 638.274171000 192.168.2.131 10.0.2.15 SIP 636 Request: INVITE sip:1002@10.0.2.1:5063 |
⊞ Frame 448: 636 bytes on wire (5088 bits), 636 bytes captured (5088 bits) on interface 1
⊞ Ethernet II, Src: Vmware_0d:30:dc (00:0c:29:0d:30:dc), Dst: Vmware_ec:bc:b0 (00:50:56:ec:bc:b0)
⊞ Internet Protocol Version 4, Src: 192.168.2.131 (192.168.2.131), Dst: 10.0.2.15 (10.0.2.15)
⊞ User Datagram Protocol, Src Port: 5063 (5063), Dst Port: sip (5060)
⊟ Session Initiation Protocol (INVITE)
  ⊞ Request-Line: INVITE sip:1002@10.0.2.1:5063 SIP/2.0
  ⊟ Message Header
    ⊞ Via: SIP/2.0/UDP 10.0.2.15:5060;branch=z9hG4bK-2078-5-0
    ⊞ From:"Asterisk" <sip:1001@10.0.2.15:5060>;tag=2078Tag005
    ⊞ To: "Riddhi"<sip:1002@127.0.1.1:5063>
      Call-ID: 5-2078@127.0.1.1
    ⊞ CSeq: 1 INVITE
      Max-Forwards: 70
      Content-Type: application/sdp
      URL=www.serverAcertificate.com
      Allow: INVITE,ACK,CANCEL,OPTIONS,BYE,REGISTER
      User-Agent: Linphone 3.2.1
    ⊞ Hash: ec39ec0e95527bebd33824ce3174b56fad9b85e2
      Content-Length:     0
⊞ Session Initiation Protocol (SIP as raw text)
```

Fig. (6) Modified SIP INVITE from Server to Client

At this point, the generated hash is encrypted by client and able to send along with the original header to the receiver side and at receiver side, the encrypted hash is decrypted and compared with the hash value contained in the original header

## REFERENCES

[1] R.Zhang, X. Wang, X. Yang, and X. Jiang. “Billing attacks on SIP-based VoIP systems”. In WOOT ’07*:* Proceedings of the first USENIX workshop on Oensive Technologies, pages 1–8, Berkeley, CA, USA, 2007. USENIX Association.

[2] Global NGN IP VoIP - Analyses Statistics and forecasts. http://www.marketresearch.com/product/display.asp?productid=1513239&g=1 2007.

[3] D. Richard Kuhn, Thomas J. Walsh, Steffen Fries “Security Consideration for Voice Over IP Systems” National Institute of Standards & Technology. Gaithersburg

[4] Paul Stalvig “Session Initiated Protocol – A Five Function Protocol”

[5] Rakesh Arora “Voice Over IP Protocols and Standards”

[6] http://www.ietf.org/rfc/rfc3261.txt

[7] Jill Slay12, Matthew Simon1, David Irwin “Voice Over IP And Forensics: A Review of Recent Australian Work”, University of South Australia, Mawson Lakes, SA 5095, AUSTRALIA.

[8] Prof. Dr. Even Eren, Dr. Kai-Oliver Detken “Voice-over-IP Security Mechanisms – State-of-the-art, risks assessment, concepts and recommendations”, 2007.

[9] A. Nemi, J. Arkko, V. Torvinen, " Hypertext Transfer Protocol(HTTP) Digest Authentication Using Authentication and Key Agrement(AKA)”, IETF RFC 3310, 2002.