AUTHENTICATION SCHEMES USING IDENTITY-BASED CRYPTOGRAPHY

by

Yolguly ALLABERDIYEV

June 2009

AUTHENTICATION SCHEMES USING IDENTITY-BASED CRYPTOGRAPHY

by

Yolguly ALLABERDIYEV

A thesis submitted to

the Graduate Institute of Sciences and Engineering

of

Fatih University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Engineering

June 2009
Istanbul, Turkey

APPROVAL PAGE

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Assist Prof. Tuğrul YANIK
Head of Department

This is to certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assist Prof. Tuğrul YANIK
Supervisor

Examining Committee Members

Assist. Prof. Tuğrul YANIK _____

Assist Prof. Nahit EMANET _____

Assist Prof. Özgür ÖZDEMİR _____

It is approved that this thesis has been written in compliance with the formatting rules laid down by the Graduate Institute of Sciences and Engineering.

Assoc. Prof. Nurullah ARSLAN
Director

AUTHENTICATION SCHEMES USING IDENTITY-BASED CRYPTOGRAPHY

Yolguly ALLABERDIYEV

M. S. Thesis - Computer Engineering
June 2009

Supervisor: Assist. Prof. Tuğrul YANIK

ABSTRACT

After the realization of Internet, traditional telephone communication, Public Switched Telephone Network (PSTN), has been leaving its role to Voice over Internet Protocol (VoIP). This process inquires the security and performance of SIP (Session Initiation Protocol) which is standardized protocol in almost all VoIP applications. The common SIP authentication mechanism in most applications is the HTTP Digest Authentication. This mechanism is easy to implement and delivers high performance results. But the various security vulnerabilities of this authentication method forces us to search for alternatives. Identity based signature schemes have significant advantages over certificate based signature schemes. Even though various ID based authentication schemes were proposed we couldn't come across an implementation that gives us an idea about the performance costs. In this thesis we integrated two different ID based authentication schemes into a well known and widely used open source SIP proxy server and obtained real performance data. The results show that the current ID based signature schemes performance is not sufficient for the time being. But if further improved the advantages it provides could increase its visibility in SIP based applications.

Keywords: Session Initiation Protocol (SIP), Identity-based Cryptography, Identity-based signature, Performance, Security.

KİMLİK TABANLI KRİPTOLOJİYE DAYALI KİMLİK DOĞRULAMA SİSTEMLERİ

Yolguly ALLABERDIYEV

Yüksek Lisan Tezi – Bilgisayar Mühendisliği
Haziran 2009

Tez Yöneticisi: Yrd. Doç. Dr. Tuğrul YANIK

ÖZ

İnternet ve intranetlerin yaygınlaşmasıyla genel anahtarlamalı telefon ağının (PSTN) kullanıcılara sağladığı servisleri artık IP tabanlı telefon sistemleri de sunmaktadır. Genel anahtarlamalı telefon şebekesinin bazı önemli dezavantajları olması nedeni ile (mobil telefonlar) IP tabanlı telefon sistemi daha da cazip hale gelmiştir. Böylesine önemli bir servisin farkli sisteme aktarılması beraberinde birçok problemleri de getirmektedir. IP tabanlı telefon sisteminde oturum açmak için kullanılan, ve son zamanlarda standart hale gelen SIP protokolü önemle üzerinde durulması gereken bir protokoldür. Çünkü SIP protokolünün güvenli ve hızlı çalışması IP tabanlı telefon sisteminin sunduğu servisleri daha da verimli hale getirecektir. Standart SIP protokolünün güvenliği için kullanılan MD5 tek yönlü karma fonksiyonunun performansının iyi olmasıyla birlikte, son zamanlarda ortaya çıkan güvenlik açıkları standart SIP protokolünün güvenlik tehlikelerine açık olmasına neden olmaktadır. Bu tezin amacı SIP protokolünde çok daha güvenli olan Kimlik Tabanlı Doğrulama sistemini açık kaynak kodlu bir SIP proxy sunucusuna entegre etmek ve performans ölçümleri yapmaktır. Alınan sonuçlara gore Kimlik Tabanlı Doğrulama Sistemi SIP protokolünü gayet güvenli hale getirmektedir. Performanstaki düşüş ise Kimlik Tabanlı Doğrulama Sisteminin geliştirilmesiyle çözülebilir ve bu yöntem gelecekte SIP protokolünde standart olarak kullanılabilir.

Anahtar Kelimeler: Oturum Açma Protokolü, Kimlik Tabanlı Kriptoloji Sistemi, Kimlik Tabanlı Kriptolojiye dayalı İmzalama Sistemleri, Performans, Güvenlik

ACKNOWLEDGEMENT

I express sincere appreciation to Assist. Prof. Tuğrul YANIK for his guidance and insight throughout the research.

The technical assistance of Hakan KILINÇ, PhD student of GYTE, is gratefully acknowledged.

I express my thanks and appreciation to my family for their understanding, motivation and patience. Lastly, but in no sense the least, I am thankful to all colleagues and friends who made my stay at the university a memorable and valuable experience.

To my brothers

Tańyrberdi and Üseýin

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The Session Initiation Protocol (SIP) is a text based application layer signaling protocol which can establish sessions between multiple parties that want to communicate (Rosenberg et al., 2002). VoIP applications are one of the fast growing applications that use the SIP protocol (Singh et al., 2005). Although the SIP protocol's flexibility and scalability inherited from the internet applications is a significant advantage, SIP messages are exposed to a variety of security threads. Snooping, modification, spoofing and denial of service attacks are some of them explained in the literature (Geneiatakis et al., 2006, Salsano et al., 2002, McGann et al., 2005).

The work conducted on SIP security mostly focuses on the authentication and key agreement issues. While there are various ideas on the authentication mechanism, the common authentication method used in applications is the HTTP Digest Authentication which is based on a shared secret (Salsano et al., 2002). The HTTP Digest Authentication delivers high performance both on the user agent and the server side because it relies on a digest algorithm (Franks et al., 1999). But on the other hand it is subject to server spoofing and password guessing attacks.

The reason why most implementations use the HTTP Digest Authentication is due to its performance. In (Salsano et al., 2002) we can see that the processing load caused by the authentication mechanism is not avoidable. Although the overhead imposed by authentication doesn't have a great effect on SIP performance under normal network conditions, the work in (Cha et al., 2007) points out that in a congested network the call setup delay can increase significantly due to the authentication overhead.

The SIP protocol needs an authentication mechanism that has a reasonable authentication overhead and avoids the security vulnerabilities the current mechanism has. It is possible to avoid the security vulnerabilities in the authentication mechanism using a certificate based authentication protocol. But a primary constraint is that the

recipients must posses or obtain the public key component to validate the signatures (Kong et al., 2006).

Another choice is to use an ID based (Identity-based) cryptosystem where the public key can be represented as an arbitrary string such as an email address, phone number or any identifying information. This mostly eliminates the need of public key and certificate management. A sender who knows the identity of its receiving party encrypts a message using receiver's identity ID_A. Unless receiver gets private key for ID_A from trusted authority, Private Key Generator, encrypted message can not be decrypted. Significant work has been conducted on ID based authentication and key agreement schemes.

N.P. Smart studied Identity based authenticated key agreement protocol using Weil pairings (Smart, 2002). On his research he pointed out the drawback of naïve Diffie-Hellman key agreement protocol. Obviously man-in-the-middle attack can succeed in the naïve Diffie-Hellman key agreement scenario. This problem can be solved with the benefits of PKI, but it brings another drawback, namely Certificate Authority. To this problem Smart suggested use of identity based authenticated key agreement protocol using Weil pairings that eliminates the Certificate authority problem and secure against man-in-the-middle attack.

Eun-Jun Yoon and his colleague proposed password based authenticated key agreement protocol using Weil pairings in three rounds (Yoon et al., 2007). Password is assumed to be shared prior to authentication process via secure channel and kept in both parties. They proved that $SAKA_{WP}$ protocol is secure against reply attack, password guessing attack, man-in-middle attack and etc.

Kyusuk Han presented secure VoIP using identity based cryptography (Han et al., 2007). Their proposed design includes use of Hess's signature algorithm (Hess, 2003) for SIP authentication and one-way two party authenticated key agreement protocol based on identity based cryptography (Okamoto et al., 2005) for SRTP (Secure Real-time Transport Protocol). Nature of identity based cryptography reduces cost of public key management and one-way key agreement with signature also reduces cost versus two-pass key agreement are notable advantages of Han et al's design.

Even though various ID based authentication schemes were proposed we couldn't come across an implementation that gives us an idea about the performance costs on

real environment. In this thesis we integrated two different ID based authentication schemes into a well known and widely used open source SIP proxy server called OpenSIPS and obtained real performance data (Voice System, 2005). We used the PBC (Pairing Based Cryptography) library implemented by Ben Lynn to realize the ID based authentication schemes (Lynn, 2005).

This thesis is organized as follows. Chapter II presents brief background information on SIP and its security mechanisms as well as attacks. Chapter III describes mathematical background preliminaries of identity-based cryptography and some hard computational problems. Chapter IV illustrates identity-based cryptography from its idea to implementation and benefits of bilinear pairings. Chapter V is simple tutorial how to use PBC library with code definitions. Chapter VI is about our test environment: well known open source SIP Proxy server OpenSIPS and open source sip stack client PJSIP (Ismangil et al., 2003). Chapter VII discusses the result of our test typically performance and security. And last chapter VIII is conclusion and further possible researches.

CHAPTER 2

SIP AND SIP SECURITY

2.1 SESSION INITIATION PROTOCOL

2.1.1 Overview of SIP Feature

Recently, short after the realization of Internet, audio and video transmission over IP based network, referred as VoIP (Voice over Internet Protocol), became main point of research area on multimedia communication. Because of some disadvantages of Public Switched Telephone Network (e.g. mobility), new idea for multimedia communication and protocol became an obligatory (Seedorf, 2007).

There are two main standards for signaling. One of them is H.323 which is developed by International Telecommunications Union (ITU) (ITU-T, 2006). H.323 is not included in this study. Complete description of H.323 can be found in (ITU-T, 2006). Another one is SIP (Session Initiation Protocol) developed by Rosenberg, et. al. and published in Internet Engineering Task Force (IEFT) documented in RFC 3261 (Rosenberg et al., 2002). The main different point between two standards is that ITU focuses on telephony and circuit-switched idea, while IETF focuses on data and packet-switched. Despite of simple difference, they both provide precisely like mechanisms for calling establishment and additional services. SIP has been used in VoIP communications and pretty well accepted by many applications. Nowadays, almost all VoIP applications are using SIP protocol.

Development of SIP started back in February of 1996. The first draft was published by Internet Engineering Task Force named “draft-ietf-mmusic-scip-00” included only one request type that was call setup request. “mmusic” is acronym for Multiparty Multimedia Session Control.

But the first publication by IETF is not familiar to us as SIP protocol what we know today. After study of 3 years and several revisions of this draft, IETF published “draft-

ietf-mmusic-scip-12” in 1999. This draft shaped as the SIP and contained six requests that SIP protocol has today. Later on, this draft was accepted as RFC 2543 (Request for Comments).

Since then, many revisions had been made for SIP and updated. The lists of SIP RFC are: RFC 3261, RFC 3262, RFC 3263, RFC 3264, RFC 3265, and RFC 3266. The last SIP RFC was published in July 2002. RFC 3266 supports IPv6 in Session Description Protocol (SDP).

2.1.2 SIP Protocol

SIP (Session Initiation Protocol) is a text-based, application-layer control (signaling) protocol for Internet Telephony that uses similar semantic to HTTP. The purpose of this protocol is to be able to make initiating, modifying and terminating the interactive user sessions that are evolved in multimedia communication. This multimedia communication can be multimedia distance conferences, distance learning, end-to-end video or voice communication, online games and etc. Figure 2.1 depicts the fundamental SIP operation where two SIP user agents communicate. User agents may be soft phone or LAN telephones. The caller sends invitation to recipient. And recipient sends 200 OK back. After communication establishment RTP (Real-time Transport Protocol) protocol is responsible for audio/video data transformation between agents.

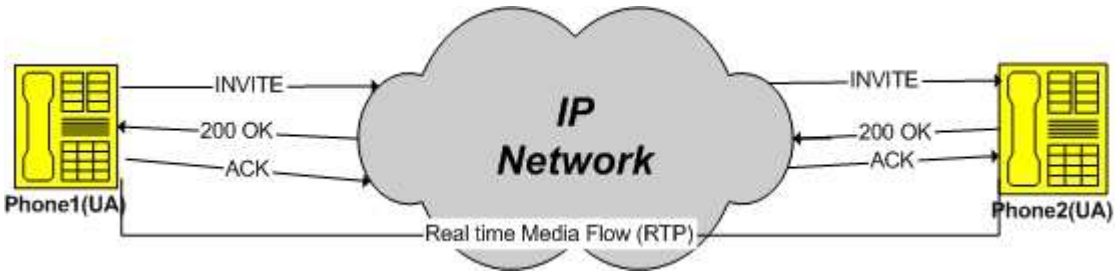

Figure 2.1 Fundamental SIP operating model

Before digging deeper of SIP, it is important to get familiar with following SIP participants and Message Formats:

User Agent: User Agent (UA) is an application program consists of User Agent Client (UAC) which is responsible for outgoing calls and User Agent Server (UAS) which is responsible for incoming calls.

Proxy Server: Proxy Server (PS) is an intermediary application that redirects requests from user agent to another end point or to another proxy server. Additionally PS supply routing, authenticating, billing functions and etc.

Registrar: Registrar Server is a server application responsible for registration of user agents and authentication as well.

Redirect Server: The purpose of Redirect Server is to inform the client that caller needs to try different route, because recipient may have changed position. Generally it happens when user agent is on movement.

There are six types of request messages (method).

Register: Message sent by client to SIP server for registration.

Invite: This message is sent from client to another client to be evolved in communication. Body of this message includes SDP (Session Description Protocol). SDP will be discussed later.

ACK: Confirm message used by caller, shows that caller has received final response from callee.

Cancel: Used to cancel request (e.g. Hang up phone)

Bye: Used by client to end the communication.

Options: Used to get information from server about its capabilities and some other optional information.

Response messages are to indicate the condition or result of request. Response messages are divided into six categories:

1×× : Means request message has been received and processing is ongoing.

2×× : Shows that request is accepted and successively completed

3×× : This request is required for next action

4×× : The request contains error. Thus request can not be completed.

5×× : Request is valid but server failed to fulfill the request.

6×× : Request cannot be accomplished by any server.

On SIP protocol end users are identified by SIP URIs (Uniform Resource Indicator). Generally user identifier takes form as sip:user@host which is identical to e-mail

address. Here, user is id of client agent, possibly name of user, and host is name of SIP service provider or domain name.

SIP Registrar Server takes role when user agent first registers. Registration message is sent by user agent to registrar. Server respond is 401 Unauthorized and sends a nonce. Nonce is a unique. It is fixed size sequence of characters. This scenario prevents replay attack and gives message freshness. User agent that receives nonce will calculate a response using nonce, own SIP URI and password, where password is received via secure channel. Server also calculates response and compares them. According to the result Server sends 200 OK message or 401 Unauthorized message again. Figure 2.2 depicts the scenario of SIP authentication.

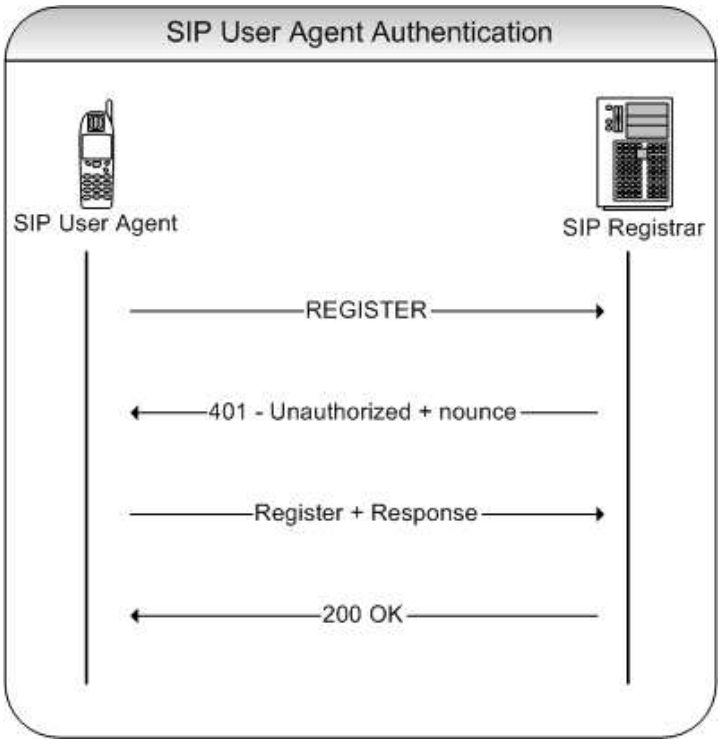

Figure 2.2 Fundamental SIP Authentication model

When caller initiates a call, invitation request is sent to locally connected SIP Proxy server. Through SIP Proxy, request is conveyed to recipient. If recipient accepts the invitation, it sends back 200 OK message and receives ACK (Acknowledge) message from caller. Figure 2.3 shows the picture of model of session initiation between two user agents.

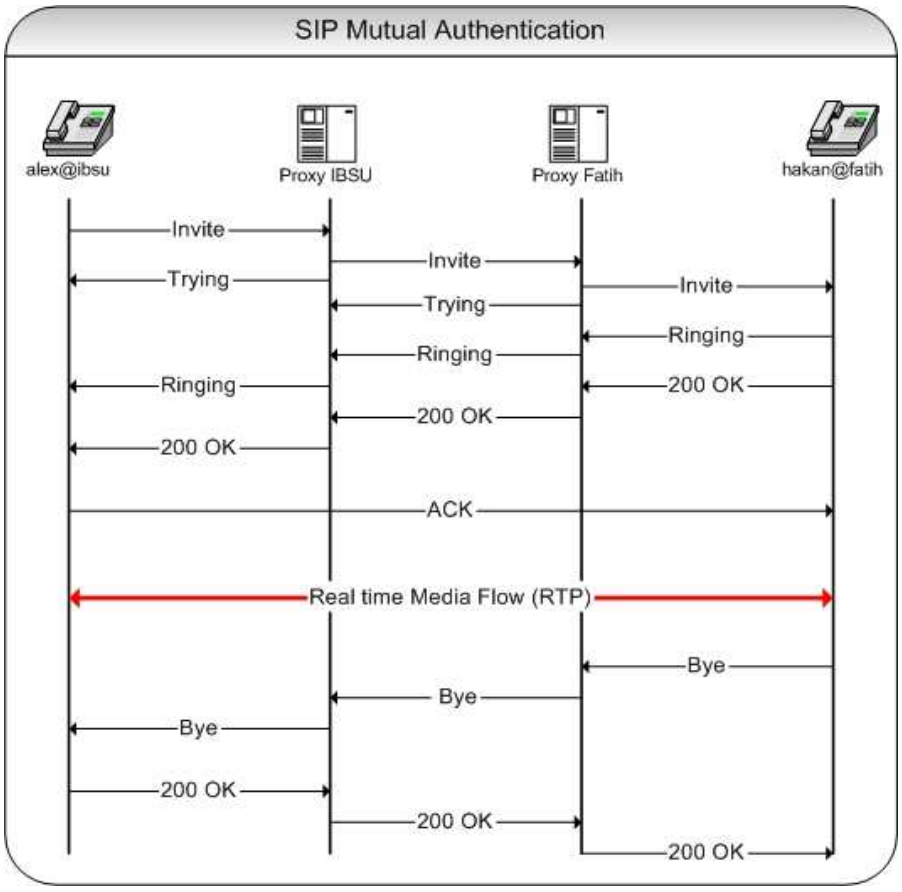

Figure 2.3 Fundamental SIP operating model

SIP is an application protocol that provides services to end users. As architecture of SIP is defined in RFC 3261(Rosenberg et al., 2002), SIP uses some features of HTTP (Hypertext Transfer Protocol) defined in RFC 2616(Fielding et al., 1999) which designates the format for web-based multimedia communication, and the SMTP (Simple Mail Transfer Protocol) defined in RFC 2821 which specifies the format of mail messages. Like HTTP and SMTP, SIP uses Internet Protocol (IP), Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) for the fundamental principles of network infrastructure.

Now, let's look inside ongoing operation in SIP. SIP is session, so it creates session, manages it as long as communication is ongoing and at the end terminates the session. These tasks may seem easy or straightforward but some complexities may arise as follows:

The first, there could be several participants (user agents in communication), thus the call would be conference (multipoint), means that the session is not end-to-end communication. Second, caller or callee or both of them may not call from same

location. They may move during communication or even session establishment. This will add requirement of holding the track of end users. Third, the media communication type is not single, rather mixture of media types. This could be text, voice/video media types. All these types have their own restraints, such as bandwidth limit, permissible transmission delay for video/voice communications.

As mentioned above SIP is an application protocol that establishes session between users. But it is important to note that SIP protocol is not responsible for media type nor communication (data) flow during conference. Thus SIP uses other protocols: Session Description Protocol (SDP) for media type and Real-time Transfer Protocol (RTP) data flow after session established. We will describe SDP and RTP in the following section.

2.1.3 Session Description Protocol

SDP is short form of Session Description Protocol defined in IETF RFC 2327 by Handley and Jacobson (Handley et al., 1998). SDP, like SIP, can be used with all transport protocols such as SAP (Session Announcement Protocol), SIP, HTTP and others. However SDP does not depend particularly on any protocol rather used conjunction with other protocols to provide full service for users. The general purpose of SDP is to inform participants about format of communication they are going to involve. On SIP scenario, SDP is included in body of SIP message. RFC 2327 notes some key points which SDP provides, are as follows:

- Name of owner (user)
- Session Time
- Session Name and purpose
- Media type(s) included in session
- Codec information
- Connection points throughout communication (port, address, format, ip version, etc.)
- Bandwidth to be used during communication and etc

As described in RFC 2327, SDP format consist of lines of text. Text from is like <*type*> = <*value*> where <type> is unique session parameter and <value> is value of corresponding parameter. SDP is composed of three main parts Session, Time and

Media. Since SDP is not main focus of this thesis, we will not go into detail. More information about SDP can be found (Handley et al., 1998).

2.1.4 Real-Time Transport Protocol

Real-time Transport Protocol (RTP) standard is defined in IETF RFC 1889 by Schulzrinne, et. al. in 1996. Recent versions are RFC 3550 and RFC 3551 (Schulzrinne et al., 2003). RTP is developed to service the delivery of end-to-end real-time data packets like interactive voice/video. RTP was primarily designed for multiparty conferences. To tell shortly, RTP conveys real-time data stream from sender to recipient. But usage is not restricted to this. Like SIP, RTP also does not depend on specific protocol, rather can be used with lots of protocols like SIP, SDP, etc. However, RTP does not guarantee on time delivery of data packets nor quality-of service, but relies on lower layer services. Generally RTP works on top of UDP protocol. Moreover RTP packets are numbered in sequence in order for recipient to reconstruct packages in correct order.

When we talk about RTP, we actually mean complete RTP protocol. Because RTP, in fact, consist of two parts:

- Real-time Transport protocol (RTP): to transport data with real-time characteristic
- Real-time Transport Control Protocol (RTCP): to monitor the quality-of-service and information about participants.

Applications that include RTP and RTCP are using different port for each. One port is for RTP, to convey data stream, and one for RTCP, to monitor the QoS (Quality-of-service). From its origin RTP and RTCP packets are not encrypted. Thus communication can be eavesdropping. If desired, packets can be encrypted to make communication secure.

More detailed information about RTP is described in (Schulzrinne et al., 2003).

2.1.5 Real-Time Transport Control Protocol

As we discussed before, RTP is just responsible for transporting data stream. Here, Real-time Transport Control Protocol (RTCP) takes role to provide all participants in session about quality of data transmission, QoS and etc. RTCP standard is defined in RFC 3550 by Schulzrinne, et. al (Schulzrinne et al., 2003).

RTCP sends reports periodically to all participants, containing reception statistics. These reception statistics include packet lost since last report, inter-arrival delay and such statistical information. Since every participant in a session sends RTCP to all participants, number of participants can be drawn thus average rates can be calculated. Receiving RTCP from all participants, user agents can control and adopt encoding algorithms, decide to let involve more participants and etc. More information about RTCP can be found in RFC 3550 (Schulzrinne et al., 2003).

2.2 SIP SECURITY

2.2.1 Security Threats and Attacks

SIP, like some other protocols, is open to some threats and attacks. By definition, threat and attack is a usage or entrance of unauthenticated adversary to the vulnerable system. Those attacks can be harmful to system or for user. To prevent system or protocol from adversary it is better to know types of attacks and make some preventions later.

Some types of threats are described as follows:

Replay Attack:

This attack is done by retransmitting genuine message to make an authorization and establish communication with the entity. Replay attack is generally done to the client-server systems. Among other attacks, replay attack is relatively easy. There are some types of replay attacks (Qiu, 2003).

Simple replay attack which is easy to apply, just eavesdrop legitimate message and send it later. Adversary succeeds if receiver accepts the message sent by adversary where adversary pretends to be genuine sender. Following picture illustrates the general replay attack scenario.

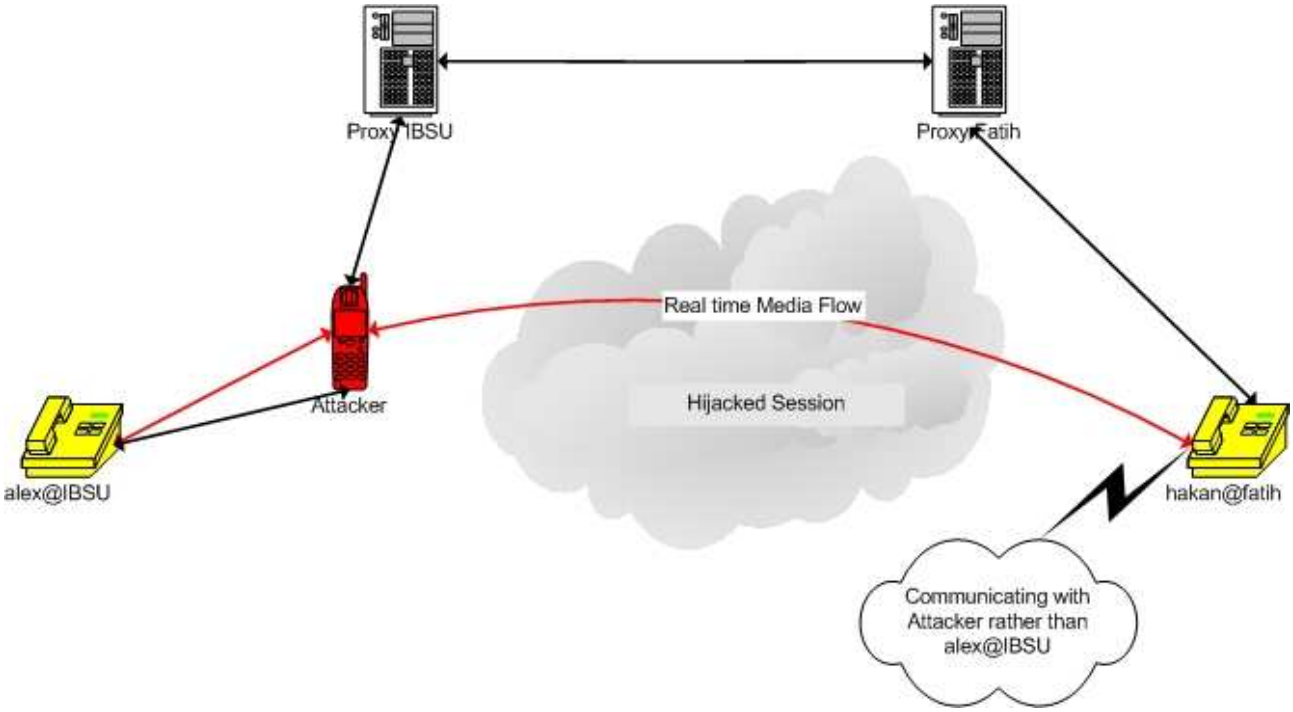

Figure 2.4 General Replay Attack scenario

Repetition that can be logged is a type of replay attack. This attack is achieved if attacker replays message in a valid interval time.

Repetition that cannot be detected called when original sender sends message and if adversary prevent the message from receiving to the recipient. Later adversary sends the legitimate message. In this case recipient can not detect that message sender is an attacker.

Registration Hijacking:

In this type of attack, rogue UA impersonates to registrar. Simply replacing own address with valid user's address that is written in From header of SIP message, attacker can succeed. As a result, valid user is seemed to be registered but in reality the registered user is an attacker. In this way attacker will be receiving all message from other connected clients in communication those are assuming to be communicating with valid user.

Proxy Impersonation:

Malicious user can pretend to be a proxy server between proxies or proxy and user. Since proxy has full control on SIP messages, successfully impersonated attacker can use all functionality of proxy server. Attacker can tear down connection, redirect to wrong user agent. Proxy impersonation attack has same role with man-in-the-middle attack described in (Stallings, 2006).

Some other attacks like Chosen Plaintext Attack, Spoofing INVITE message, Spoofing CANCEL message, Spoofing BYE messages and etc are described in (Collier, 2005) and (Qiu, 2003).

In SIP Security Mechanism, HTTP Digest Authentication is used. Following section describes HTTP Digest Authentication Mechanism.

2.2.2 HTTP Digest Authentication

Hypertext Transfer Protocol (HTTP) is an application layer protocol developed by Fielding, et al. in IETF, documented as RFC 2616 in 1999 (Fielding et al., 1999). HTTP is used for distributed, collaborative, hypermedia information systems. In RFC 2616 “HTTP/1.1” is described.

HTTP is challenge/response standard for client-server connection, uses port 80 (by default). A user agent, which is client, request from server. Server creates a HTML files and images to response back to client. This scenario is without any authentication. Thus any client initiates correct request messages to server, will get response. To make restriction further developments are added some features in RFC 2617 (Franks, 1999). There are to types of HTTP authentication mechanisms, Basic and Digest authentication mechanisms are described below.

- **HTTP Basic Authentication**

HTTP Basic Authentication as described in RFC 2617 (Franks, 1999) provides client to enter credentials in the form of – user name and password – to prevent unauthenticated users to use resources. Challenge/response process between client and server goes as follows:

Client sends request message to server. Server responds back 401 response code. At this

stage client may cancel the connection. Later on, client concatenates user name and password with ":" between them. Resulting string is encoded with Base64 algorithm (Linn, 1987) and sent to the server. Encoding password and username makes unreadable by naked eye. However, Base64 algorithm does not guarantee strong security. Encoded result can be decoded easily with zero knowledge. As a result any type of attacks like Replay attack and etc. can be successful. Because of cleartext password can be sniffed easily, HTTP basic authentication is deprecated from SIPv2.0. Result of weak security basic authentication, new authentication scheme called HTTP Digest authentication developed. Next section illustrates digest authentication.

- **HTTP Digest Authentication**

Unlike Basic Authentication, Digest Authentication documented in RFC 2617 (Franks, 1999), provides encoding of password and username plus nonce which is random n-length string used for once using MD5 algorithm. Figure 2.5 depicts HTTP digest authentication scheme used in SIP authentication.

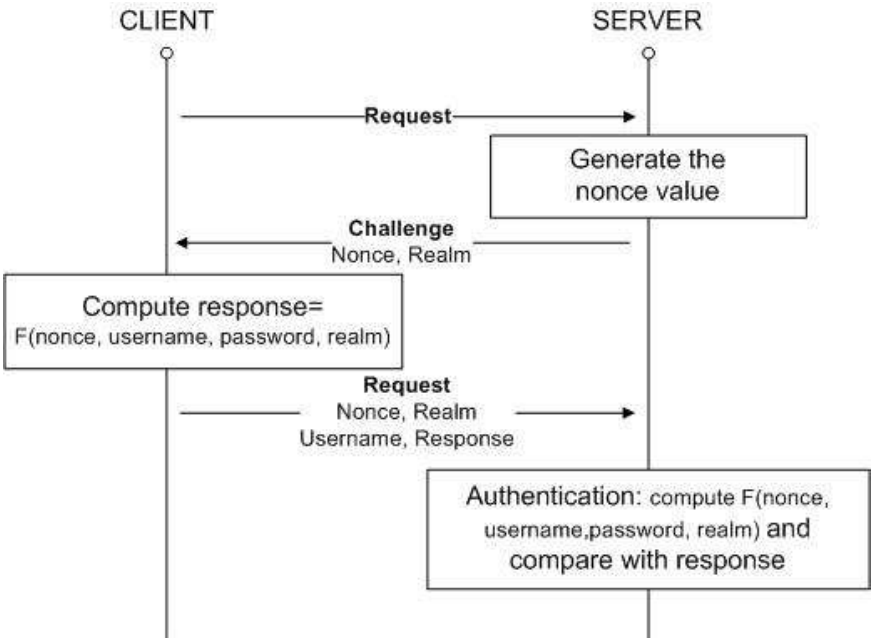

Figure 2.5 HTTP digest authentication scheme

No transmission of password makes it much more secure than Basic authentication.

MD5 algorithm designed by Ronald Rivest in 1991, documented in RFC 1321 (Rivest, 1992) is known as cryptographic hash function with 128-bit hash value and well accepted by many security applications and commonly used to control the integrity of files. Also known as one-way function means that it is difficult to get original input when only output is known.

HTTP Digest Authentication is accepted as standard in SIP protocol for its security mechanism (Rosenberg et al., 2002). However, digest authentication has vulnerability against brute force attacks (Wang, 2005). Using Rainbow tables (dictionary) weak passwords can be detected by matching the result of MD5. Thus, for SIP agents, weak passwords are not advised. Combination of letters with numbers and some characters are difficult to find out password but not practical for users.

CHAPTER 3

MATHEMATICAL BACKGROUND

3.1 CONTRIBUTION OF MATHEMATICS

Recent decades shows that mathematic is taking important role in the field of engineering especially in cryptography because of its powerful, reliable and robust solutions to the problems. From the very beginning in cryptography there are several forms of encryption/decryption. Classical Encryption mentioned as first type consist of several important encryption/decryption techniques. Substitution, Transportation, Rotor Machines and etc are first techniques used in cryptography. However those techniques used for several years, they did not go long because of their simple scheme and without usage of powerful features of mathematics. If we continue to tracing the history of cryptographic techniques we meet DES (Data Encryption Standard) and later on AES (Advanced Encryption Standards) are still in use. Their schemes are much more complex. Asymmetric cryptography is recently founded and realized using algebra of mathematic.

In order to understand underlying mathematics of those cryptographic functions, some topics in algebra needed to be clarified. Ongoing subtitles describe preliminaries.

3.2 MODULAR ARITHMETIC

Let q be a positive integer. We denote a set of $Z_q = \{0, \ldots, q-1\}$

Assume two numbers x and y where y is element of Z_q. If $x = y \pmod q$ that means x is different from y by i multiple of q, we say that x and y are congruent modulo q. From here we can say that every integer x has some congruent $y \in Z_q$ and y is called residue.

Example:

$39 = ? \pmod 7$

39 = 32 = 25 = 18 = 11 = 4 (mod 7). There are infinite numbers which is congruent 4 modulo 7.

Arithmetic**:**

All arithmetic operations can be calculated in two ways. First, make arithmetic operation get result and take mod. Or get mod of each operand first, later do arithmetic.

Example:

10*5 = 50 = 2 (mod 4)

or

10 = 2 (mod 4) and 5 = 1 (mod 4), then 2*1= 2 (mod 4)

All arithmetic operations can be used on modulo as given on the example above, except division.

Division does not treat like other operations.

Let's say, 15 = 5 (mod 10). If we divide both sides by 5, we get 3 = 1 (mod 10) which is not correct. Thus the meaning of division in modular arithmetic is not same idea as with normal arithmetic division.

Inverse:

For each element $x \in Z_q$, if we can find $y \in Z_q$ such that x*y = 1 (mod q) then y is inverse of x modulo q. If inverse does not exist then it is undefined.

Example:

3 * 7 = 1 (mod 10). So 7 is inverse of 3 and vise verse on modulo 10.

2 * ? = 1 (mod 10). Inverse of 2 does not exist. That's why result is undefined.

Let's come back to the division problem. Instead of dividing number, we can multiply with its inverse, if exist.

x and $y \in Z_q$, then $x/y = x*y^{-1}$ (y^{-1} is in inverse of element y)

5 / 7 = ? (mod 10).

Since $7^{-1} = 3$ (mod 10), we can multiply by 3 instead. 5/7 = 5 * 3 = 15 = 5 (mod 10)

3.3 MODULAR EXPONENTIATION

Assume that we are given a problem to find out $4^{10} = x$ (mod 11). To calculate 4^{10} and finding x modulo 11 is called modular exponentiation.

One way to do this is that we can calculate multiplying 4 by itself 10 times and reduce

result to modulo 11. But this takes time and hard to compute. What, if we try in the following way.

$4^2 = 5 \pmod{11}$, $(4^2)^2 = 5^2 = 25 = 3 \pmod{11}$, $(4^4)^2 = 3^2 = 9 \pmod{11}$

Thus, $4^{10} = 4^8 * 4^2 = 9 * 5 = 45 = 1 \pmod{11}$

In second way we use only 4 modular multiplication instead multiplying 4, 10 times, by itself.

3.4 GROUP THEORY

Additive Group:

Definition: A group (G, +) consist of set G with binary operation + on G. Additive group satisfies the following axioms.

(i) The group operation is *closure*. That is, $\forall$ a, b $\in$ G, a+b $\in$ G

(ii) The group is *associative* : $\forall$ a, b, c $\in$ G, a+(b+c) = (a+b)+c

(iii) Additive *Identity*. There exist an element 0 such that a+0 = 0+a = a, $\forall$ a $\in$ G

(iv) Additive *Inverse* element. For each a $\in$ G there exist b $\in$ G, called inverse of a, such that a+b = b+a = 0

If a group satisfies axiom (v), then group is called Abelian or Commutative group.

(v) A group is *abelian (commutative)* if, a+b = b+a, $\forall$ a, b $\in$ G

Definition: The number of elements in a group G is called *order of G*, denoted by |G| or ord(G).

Example:

G = {1, 4, 6, 9}, the order of G is 4 (|G| = 4 or ord(G) = 4)

Definition: A group G is called *cyclic* if there exist an element *a*, such that for all $k \in G$, there exist number *i* that satisfies condition $a^i = k$. And an element *a* is called *generator* of group.

Example:

The set of positive integers Z^+ is not a group because of no inverse elements. But set

of integers Z forms group and abelian group under addition operation with identity element 0.

Multiplicative Group:

A group (G, *) consist of set G with binary operation * G satisfying the following axioms.

(i) *Closure* under multiplication: $\forall$ a, b $\in$ G, a*b $\in$ G

(ii) *Associative* under multiplication: $\forall$ a, b, c $\in$ G, a*(b*c) = (a*b)*c

(iii) *Identity* element: : $\forall$ a $\in$ G, there exist an element 1 such that, a*1 = 1*a = a

(iv) *Inverse* element: : $\forall$ a $\in$ G, there exist an element b, b is called inverse of a, such that a*b = b*a = 1

Any group satisfies the axioms above are called *group under multiplication (multiplicative group).*

Example:

A group G = Z_5 = {0,1,2,3,4} is group under addition but it is not group under multiplication since all elements has no inverse. However, G will be a multiplicative group if we eliminate element 0. $Z_5/\{0\}$ is a group under multiplication.

3.5 ELLIPTIC CURVE

Elliptic curve is a plane curve defined by Weierstrass equation as follows.

$y^2 = x^3 + ax + b$ (General elliptic curve formula : $y^2 = Ax^3 + Bx^2 + Cx + D$)

Here, *(x,y)* is a point on an elliptic curve while *a* and *b* are real numbers. Another requirement of an elliptic curve is that the curve must be *non-singular*. Geometrical meaning of non-singular is that there is no self-intersection of curve, while algebraically curve must satisfy the equation $4a^3 \neq -27b^2$. Generally we can define set of elliptic curve $E = \{(x, y) : y^2 = x^3 + ax + b\} \cup \{O\}$ where O is the *point at infinity*, going to be mentioned on the next section.

Following picture shows the general view of an elliptic curve.

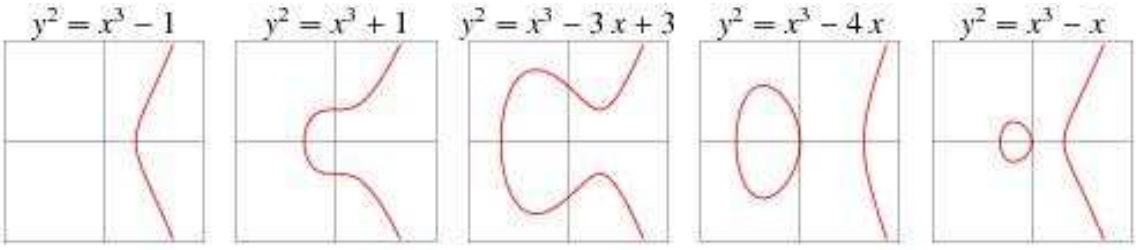

Figure 3.1 Elliptic Curve (http://mathworld.wolfram.com/EllipticCurve.html)

Elliptic Curve Group:

Till now we talked about elliptic curve on real numbers. The condition is not useful for cryptographic protocols where infinite numbers are used. So, we need to think elliptic curve over some finite filed F_q.

Example:

Now, let's say we defined an elliptic curve $y^2 = 2x^3 + 4x$ over Z_5. Both coefficients and points (x, y) are modulo 5 and can take values {0, 1, 2, 3, 4}.

for x=0 $\Rightarrow$ $y^2 = 0$, y=0 one solution (0,0) (mod 5)

for x=1 $\Rightarrow$ $y^2 = 1$, y=1, 4 two solutions, (1,1) and (1, 4) (mod 5)

for x=2 $\Rightarrow$ $y^2 = 4$, y=2, 3 two solutions, (2, 2) and (2, 3) (mod 5)

for x=3 $\Rightarrow$ $y^2 = 1$, y=1, 4 two solutions, (1,1) and (1, 4) (mod 5)

for x=4 $\Rightarrow$ $y^2 = 4$, y=2, 3 two solutions, (2, 2) and (2, 3) (mod 5)

As a result the set of elliptic curve $y^2 = 2x^3 + 4x$ modulo Z_5 is

$E(Z_5)=\{(0,0), (1,1), (1,4), (2,2), (2,3)\} \cup \{O\}$. In this example we did not meet any no-solution condition, like $y^2 = 3$. For those situations we define element *O*, point at infinity.

Elliptic Curve forms a group with following features over filed F_q.

1. $\forall P, Q \in E$ then $P + Q \in E$ (closure)
2. $\forall P \in E \quad P + O = O + P = P$ (O: identity element)
3. $\forall P \in E$, $\exists Q \in E$ such that $P + Q = Q + P = O$. Q is called inverse of P (Q = -P)
4. $\forall P, Q, R \in E \quad P + (Q + R) = (P + Q) + R$ (associative)

5. $\forall P, Q \in E, \quad P + Q = Q + P$ (abelian)

Elliptic Curve Point Addition:

In an elliptic curve we can do arithmetical operations on points. Addition of two points results another point on an elliptic curve. Adding a point by itself *n* times, that is *n*P*, can also be calculated easily. In previous sections, I mentioned about element *O*, point at infinity. Point at infinity means there is no any third intersection point on an elliptic curve when we add, draw a straight line over, two points. Because there is no intersection point, it is called point at infinity and included in a set in order to consider addition of two opposite points. Opposite points are points having same x value but opposite sign of y. Assume point P(2, 7) where its opposite point is –P(2, -7).

Now let's look at the Figure 3.2.

On picture number 1, P and Q are points. If we draw a straight line along P and Q, line will intercept curve on third point R. But result is not R. The result is –R, that is, $P + Q = -R$.

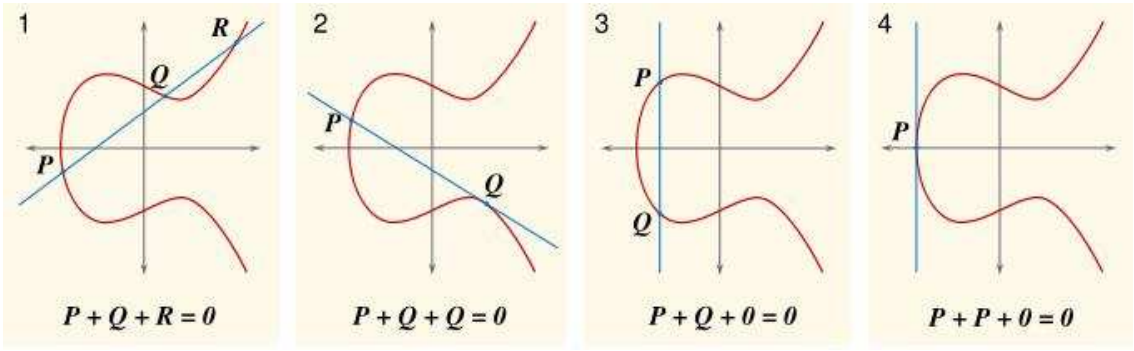

Figure 3.2 Elliptic Curve Point Addition
(http://upload.wikimedia.org/wikipedia/commons/c/c1/ECClines.svg)

When we draw a tangent line on point P and if line intercept curve on second point, we will get result P+P. Drawing tangent line to the resulting point is P+P+P and can be calculated 3*P. This is called *point exponentiation* which is obviously easier than modular exponentiation of real numbers.

All calculations we have done yet are related with geometrical representation of elliptic curves. There are *algebraic formulas* as follows:

Assume P, Q and R $\in$ *E*. Draw a line $\overline{PQ}$: y = cx + d, $P(x_1, y_1)$, $Q(x_2, y_2)$ and $R(x_3, y_3)$

Let elliptic curve E be given by : $y^2 = x^3 + ax + b$

When $P \neq Q$

$$c = \frac{y_2 - y_1}{x_2 - x_1}$$

$$x_3 = c^2 - x_1 - x_2$$

$$y_3 = c(x_1 - x_3) - y_1$$

$$R(x_3, y_3)$$

When $P = Q$

$$c = \frac{3x_1^{\,2} + a}{2y_1}$$

$$x_3 = c^2 - 2x_1$$

$$y_3 = c(x_1 - x_3) - y_1$$

$$R(x_3, y_3)$$

We keep description of elliptic curve short. Because detailed description is not in our scope. More information about elliptic curves are described in (Rosen, 2006).

3.6 BILINEAR MAP

In general, a bilinear is mapping of two arguments that is linear in each. If we focus on elliptic curve, pairing is mapping of two points in an elliptic curve to an element of multiplicative group of a finite field. The aim of this section is to describe bilinear function which will be mentioned in Chapter 4 while discussing Identity Based Cryptography that uses most well known implementations of pairings –Weil and Tate pairings (Washington, 2008).

Assume an elliptic curve $E(F_q)$

A point $P \in E(F_q)$, where order of P is q and generates additive group G_1, that is $< P >= G_1$. G_2 is a multiplicative group of order q. So both groups have same order q.

Now, bilinear function is:

$$e : G_1 \times G_1 \longrightarrow G_2$$

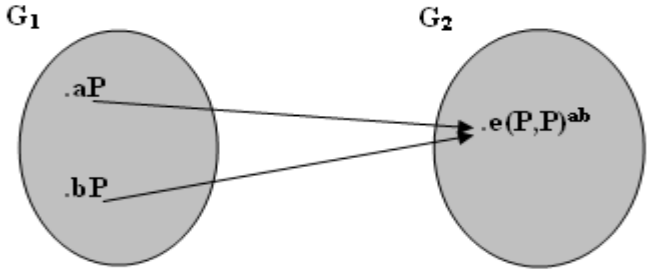

Figure 3.3 Bilinear Pairing

Bilinear mapping has three properties:

1. Bilinearity:

$$\forall P, Q \in G_1 \text{ and } \forall a, b \in Z_q^*$$

$$e(aP, bQ) = e(bP, aQ) = e(P, Q)^{ab}$$

In other words, $\forall P, Q, R \in G_1$

$$e(P + R, Q) = e(P, Q)e(R, Q)$$

and

$$e(P, R + Q) = e(P, R)e(P, Q)$$

2. Non-Degeneracy:

$$\forall P \in G_1, P \neq 0 \Rightarrow e(P, P) \neq 1$$

In other words,

$< e(P, P) >= G_2$, if $e(P, P)$ is generator of G_2 then function e is called *admissible bilinear* function. From now on we implicitly mean admissible bilinear when map we say bilinear map.

3. Computability: e is efficiently computable

Group G_1 and G_2 that hold properties above can be constructed. Weil and Tate pairings

prove the existence of such groups.

Up to now, we did not talk about self-bilinear map on bilinear map. Self-bilinear mapping would be much more powerful. But there is no practical solution yet for self-bilinear mapping and still remains open problem.

Self-bilinear map, when $G_1 = G_2$

$$e : G_1 \times G_1 \longrightarrow G_1$$

3.7 HARD COMPUTATIONAL PROBLEMS

Computationally hard problems plays essential role in the field of cryptography. Almost all latest cryptographic methods, especially those constructed after the implementation of asymmetric cryptography, use one of hard computational formulas for their underlying security issues. Those problems are mentioned in the following subtitles without proof.

3.7.1 Factorization Problem

By the fundamental theorem of arithmetic, every integer greater than 1 has unique prime factorization. However, it does not give any practical solution to obtain prime factors. Integer factorization is to find prime factors of composite number. When they are multiplied, we get original composite number.

Knowing n it is difficult to find p and q where p, q are large prime numbers.

$$n = p \times q$$

Example:

$555 = 3 \times 5 \times 37$ (3, 5 and 37 are prime numbers)

$100 = 2^2 \times 5^2$ (2 and 5 are prime numbers)

It may seem easy for small numbers. For large number like 160 bits no practical solution yet.

RSA cryptography (Stallings, 2006) uses factorization problem for its security and still in use.

3.7.2 Discrete Logarithm Problem (DLP)

Discrete Logarithm applies on mathematical structure group. Generally this group is multiplicative cyclic group Z_q^* with generator g. *Discrete exponentiation* is to find out $x \in F_q^*$ by calculating $g^n \pmod q$.

Example:

Let's compute 7^3 on group Z_{19}^*. $7^3 \equiv ? \pmod{19}$

$7^3 \equiv 49 * 7 \equiv 11 * 7 \equiv 1 \pmod{19}$

Discrete Logarithm is inverse operation of *discrete exponentiation.*

Given g, q and x, find n ?

$$g^n = x \pmod q \ \rightarrow \ n = d \log_g x$$

Finding n is believed to be difficult and hard direction of one-way function. Thus DLP is used in several public key cryptography including ElGamal and Digital Signature Standard (DSS) (Stallings, 2006).

Example:

$7^n = 8 \pmod{19}$, find n?

$7^4 = 7^2 \times 7^2 = 11 \times 11 = 7 \pmod{19}$ n is not just 4 but

$7^4 = 7^7 = 7^{10} = ... = 7 \pmod{19}$.

3.7.3 Elliptic Curve Discrete Logarithm Problem (ECDLP)

Given elliptic curve $E(F_q)$: $y^2 = x^3 + ax + b$ and $P, Q \in E(F_q)$

$\exists n$ such that $nP = Q$

Knowing P and Q, it is computationally difficult to find n unless number of points on elliptic curve E over F is not same as number of elements in F. It is because point counting is important while selecting elliptic curve over field F.

Size of elliptic curve determines its resistance against attacks. However, elliptic curve provides same level of security as RSA with smaller group size. MOV reduction demonstrates reduction of ECDLP to DLP (Menezes et al., 1993).

3.7.4 Computational Diffie-Hellman Problem (CDH)

CDH is related with Diffie-Hellman assumption. Consider multiplicative cyclic group $G =< g >$ of order q and $a, b \in Z_q$.

Given g^a and g^b, it is computationally intractable to compute g^{ab}. This hard problem is related with diffie-hellman problem on real numbers used in Diffie-Hellman Key Agreement protocol (Rescorla, 1999).

3.7.5 Decisional Diffie-Hellman Assumption (DDH)

DDH was proposed by Dan Boneh in 1998 (Boneh, 1998a). He showed that DDH is stronger than discrete log problem.

Consider multiplicative cyclic group G order q with generator g. DDH assumption states that given g^a, g^b where $a, b \in Z_q$ resulting g^{ab} is also some random element from G. DDH has strong dependency to discrete log problem. If there exist an algorithm calculating $d\log_g x$ then DDH is no longer hard to break.

3.7.6 Bilinear Diffie-Hellman Assumption (BDH)

Assume multiplicative group G with generator g from an elliptic curve E.

Given g, g^a, g^b, g^c where $a, b, c \in Z_q$, then computing $e(g,g)^{abc}$ is computationally difficult. This problem is also assumed to be hard problem.

CHAPTER 4

IDENTITY BASED CRYPTOGRAPHY

4.1 DEFINITION

Obviously most powerful cryptosystem that has been used recently is asymmetric cryptography, also named as public-key cryptography (Stallings, 2006). In public-key each user owns two, public and private keys, generated by himself/herself or central trusted third party using some secure channel for key transfer. Public key P_A of user A is generated from secret key (private), S_A, of user A using one-way function. In public key cryptography public key of all users are public while secret keys are kept secret. One-way function must be computationally infeasible to compute secret key from public key incase any adversary attempts to recover the secret key from public.

In order to send encrypted message M to receiver Alice, sender Bob uses receiver's public key, that is P_A public to all, $C = E(P_A, M)$. Ciphertext C is send to Alice through insecure channel. Assume adversary Emma captured the plaintext. Since message was encrypted using Alice's public key, message can only be decrypted by Alice's private key. Thus Emma could not get any useful information from captured ciphertext. Receiving ciphertext from Bob, Alice decrypts message using her private key, $M = D(S_A, C)$ as shown in Figure 4.1.

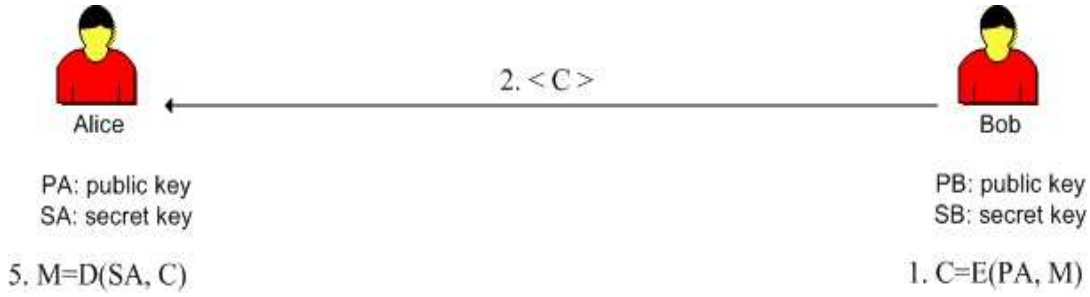

Figure 4.1 Public-key cryptography message enc/dec

Until know public-key cryptography looks like perfect. Everybody knows each others public key that enables user to send secret message to someone else and all private (secret) keys are kept secret. But what if user Emma pretends to be Alice and what if she publishes her public key as if it is Alice's? Here problem arises. Emma can eavesdrop all messages sent to Alice. As a result sender must use authenticated public key, otherwise it may be a trap.

Public-key infrastructure (PKI) (Stallings, 2006) is conventional solution to the drawback of public-key cryptosystem. In PKI system, central trusted party called *Certification Authority* is added. CA plays critical role in PKI. All authenticated public keys are stamped by CA. However PKI is a good solution, authenticating to the CA brings new concern to the PKI environment.

In 1984, Adi Shamir (Shamir, 1984) proposed the idea of *identity-based* cryptography (IBC). As its name represents, identity of user is used as authenticated public key. Knowing identity of user is enough to send secret message through insecure channel. IBC is like public-key infrastructure without its drawbacks. Private-Key Generator (PKG) included as fully trusted third party in IBC which gives secret key of users that is generated from user's public key (identity).

The different point on key pairs generation between conventional public-key and IBC is that in conventional public-key user generates his/her secret key and applying one-way functions gets public key while in identity-based user selects public key, typically user name, phone number, email address, social security number, ip number or some other information represents identity of user, and authenticated private key, generated by PKG, transferred via secure channel to the corresponding user. One of disadvantages of IBC is that PKG must be fully trusted since PKG itself generates private keys of all users. So, in IBC Private Key Generator is assumed to be fully trusted party.

Following section discusses history of IDC from idea to implementation and some other research area.

4.2 HISTORY

Identity-based idea was proposed by Adi Shamir (Shamir, 1984) in 1984. This new cryptographic paradigm was to get rid of drawbacks of PKI. In this idea user's identity information such as email address or phone number used as authenticated public key without need of Certificate Authority (CA). As a result new idea diminishes the complexity and burden of managing the PKI. Shamir was able to construct only identity-based signature (IBS) using RSA (Rivest et al., 1978) function. Construction of identity-based encryption (IBE) scheme became last long open problem till 2001.

In 2001 two different solutions were proposed. Boneh and Franklin (Boneh et al., 2001) implemented identity-based encryption (IBS) based on weil pairing. Cocks (Cocks, 2001) also implemented IBE scheme using quadratic residue rather than weil pairing. Thanks to their successful realization of IBC. Even though Cocks successfully implemented IBE using quadratic residue, later research on IBC uses Boneh and Franklin scheme because of large transaction on Cocks.

Realization of IBE based on weil pairings was the beginning of flourishing of identity-based cryptography. At the same year in 2001 Boneh et al (Boneh et al., 2001) accomplished implementation of identity-based signature scheme, known as *short signature* which was fundament for Boldyreva to design threshold and blind signature schemes (Boldyreva, 2003). One another two signature schemes that are used in this thesis is Cha and Cheon's IBS scheme (Cha, 2003) and Hess's scheme (Hess, 2002) based on weil pairing.

There are some works on non-identity based scheme using bilinear pairing. One of them is Joux's Tripiartite Key Agreement protocol (Joux, 2000). Joux proved that Diffie-Hellman Key Agreement protocol can be done in one round using bilinear pairing. That is, three users can agree on a key in one round using benefits of bilinear pairing.

Most recent research on identity-based cryptography is *signcryption scheme*. The main idea of signcryption is rather than singing and encrypting separately, sign and encrypt simultaneously. It may seem that no difference, but implementation shows that signcryption scheme is much faster that conventional one. Signcryption is not brand new idea that comes with identity-based cryptography.

Malone-Lee (Malone-Lee, 2003) implemented signcryption scheme using pairings respectively in 2003.

4.3 WEIL PAIRING

Weil pairing is mapping from pairs of two points on an elliptic curve $E(F_q)$ to finite field F_q. More precisely, it establishes an isomorphism between a group $< P >, P \in E(F\ )$ of order k and the k^{th} roots of unity. From cryptographic perspective, it is a mapping of ECDLP to the DLP in the extension field F_{q^k}. If k is not too big, then DLP is solvable. Since ECDLP depends on DLP, we can not talk about security of ECDLP while DLP is not secure enough.

m-Torsion point E[m] is a group of points of order m where all points $P \in E(F_q)$.

Now, let's define weil pairing.

Let k be an integer relatively prime to q. Then Weil Pairing is function:

$$e_k : E[k] \times E[k] \rightarrow F_q,$$

where q is prime or some prime power p.

Weil Pairing has some great features as follows.

- Identity: $\forall P \in E[k], e_k(P,P) = 1$
- Bilinearity: $\forall P,Q,R \in E[k], e_k(P+Q,R) = e_k(P,R)e_k(Q,R)$ and

  $$e_k(P,Q+R) = e_k(P,Q)e_k(P,R)$$

- Non-degeneracy: $\forall P \in E[k], e_k(P,Q) = 1$ for all $Q \in E[k]$ iff $P = O$ (O : identity element)
- Computable: for all $P,Q \in E[k]$, $e_k(P,Q) \in F_q$ is easily computable.

Here we defined weil pairing in short. But main point of weil pairing is clarified. Of course detailed information will help to understand deep mathematics of function that is defined in (Eisentrager et al., 2003).

4.4 IDENTITY BASED ENCRYPTION SCHEME

In 2001, Dan Boneh and M. Franklin from Stanford University successfully implemented last long open identity-based encryption (IBE) problem using weil

pairings (Boneh et al., 2001).

Before going to the detailed Boneh and Franklin IBE scheme we will figure out the general scenario of IBE scheme.

Identity based cryptography environment includes Trusted Third Party called Private Key Generator (PKG). Assume there are two users Alice and Bob where Alice wants to send secure message to Bob via insecure channel. Procedure will be outlined in the following 4 steps. Figure 4.2 illustrates the scenario.

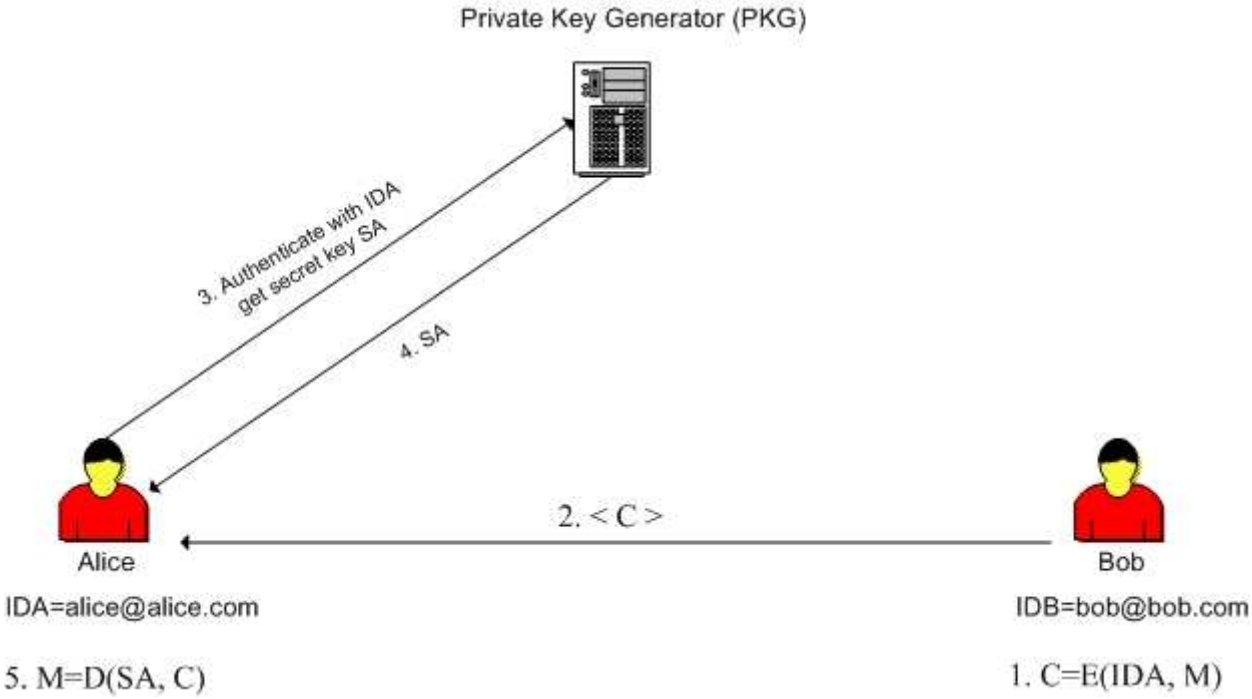

Figure 4.2 IBE scheme

- Setup:

  Private Key Generator creates its master (secret) and public keys which we call S_C and P_C respectively. P_C is delivered to all parties (users) while S_C carefully kept secret. Any adversary could get S_C will be able to eavesdrop all messages of all users who got their secret key from this PKG.

- Private Key Extraction:

  Receiver Bob authenticates himself to the PKG with his user name "bob@bob.com". PKG generates Bob's secret key S_B using P_B and own secret key S_C.

- Encryption:
  In order to send encrypted message M to Bob, Alice uses Bob's public key P_B. Obtained ciphertext C sent to Bob.
- Decryption:
  Receiving ciphertext C from Alice, Bob uses his own secret key S_B to recover the message M.

We showed general scenario of IBE above. Now let's look in detail to Boneh and Franklin scheme.

In the setup stage PKG defines a group G_1 where $< P >= G_1$ and bilinear pairing $e: G_1 \times G_1 \to F$. PKG also specifies two hash functions $H_1 : \{0,1\}^* \to G_1$ and $H_2 : F \to \{0,1\}^l$ where *l* represents the length of message. PKG select its own secret key selecting randomly $S_C \in Z_q$ where *q* is order of both G_1 and F. Then publishes its public key $P_C = S_C P$, description of group G and F, and hash functions H_1 and H_2.

Bob authenticates himself to the PKG with identity name ID_B = bob@bob.com and gets secret key $S_B = S_C Q_B$ where $Q_B = H_1(ID_B)$.

Now sender Alice can send encrypted message M to Bob. Alice calculates the ciphertext as follows:

Select random $r \in Z_q$ and compute $U = rP$ where $U \in G_1$.

Then compute $V = H_2\left(e(Q_B, P_C)^r\right) \oplus M$, typically $Q_B = H_1(ID_B)$.

After computing U and V as above, Alice sends <U,V> pair to Bob. Receiving <U,V> pair, Bob follows the decryption procedure like:

$M = V \oplus H_2\left(e(Q_B, P_C)^r\right)$

V is sent by Alice, so we need to calculate $e(Q_B, P_C)^r$. From bilinearity property

$e(Q_B, P_C)^r = e(Q_B, S_C P)^r = e(S_C Q_B, rP) = e(S_B, U)$

$e(S_B, U)$ can be calculated easily only and only by Bob. Because S_B is secret key that is only known by Bob. As a result only Bob can decrypt the ciphertext.

Boneh and Franklin scheme was proven to be secure against plaintext attack in the random oracle model (hash functions are assumed to be ideal hash function) and BDH problem is computationally hard.

One disadvantage of IBE scheme is that all workloads are burden on a single PKG. To solve this problem Hierarchical IBE scheme was proposed by Horwitz and Lynn (Horwitz et al., 2002).

4.5 IDENTITY BASED SIGNATURE SCHEME

Obviously IBS scheme goes like IBE scheme except user signs the message with his/her own secret key. In the following 4 steps Alice attempts to send signed message to Bob. As mentioned in 4.4, IBS scheme as has same procedure and signing goes as follows:

Figure 4.3 illustrates the IBS scheme.

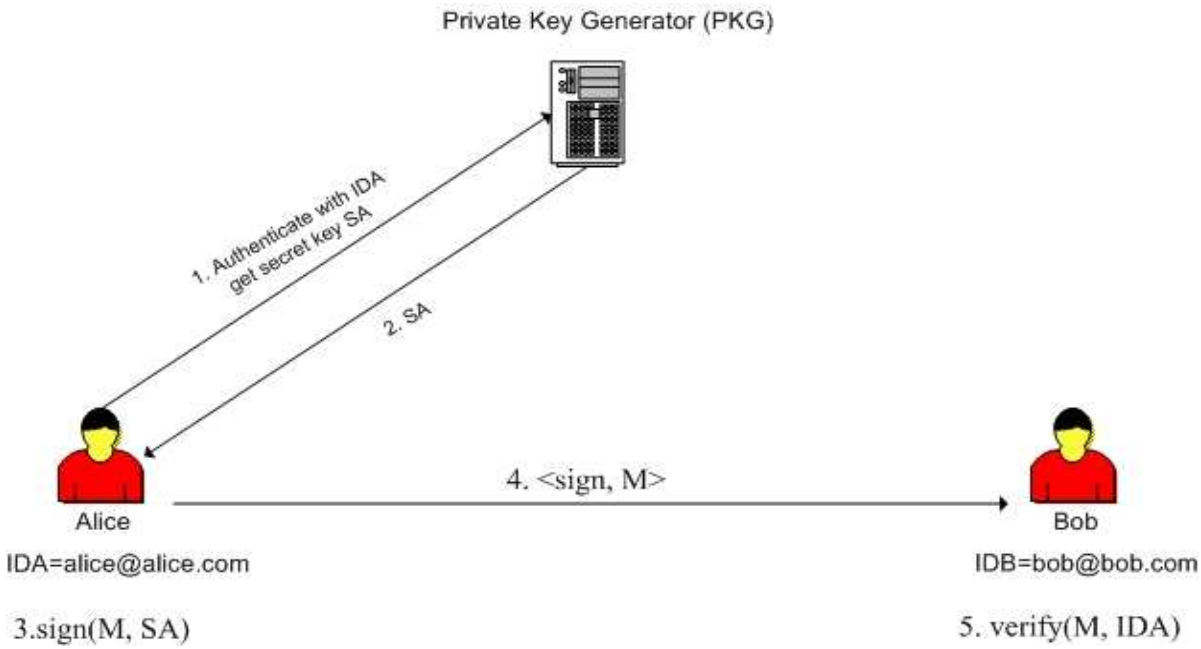

Figure 4.3 IBS scheme

- Setup:

  Private Key Generator creates its master (secret) and public keys which we call S_C and P_C respectively. P_C is delivered to all parties (users) while S_C carefully kept secret. Any adversary could get S_C will be able to eavesdrop all messages

of all users who got their secret key from this PKG.

- Private Key Extraction:

  Sender Alice authenticates himself to the PKG with his user name "alcie@alice.com". PKG generates Alice's secret key S_A using P_A and own secret key S_C where $P_A = H_1(ID_A)$

- Signing:

  In order to send signed message M to Bob, Alice uses her private key S_A. Later concatenation of signature δ and message M sent to Bob.

- Verification:

  Receiving message M and signature δ from Alice, Bob uses Alice's public key P_A and PKG's public key P_C to verify the message M.

There are several implementation of IBS scheme based on bilinear pairings. We will not describe all of them but three well known scheme BLS. Hess and Cha & Cheon's signatures will be described in the next chapter.

BLS Short Signature:

In 2001, immediately after the realization of IBE scheme based on weil pairings D. Boneh, B. Lynn, and H. Shacham designed short IBS scheme, named as *BLS* or *BLS short signature* (Boneh et al., 2001). Security is proven under random-oracle model against chosen message attack depending on CDH is hard on certain elliptic curve over finite field characteristics.

Signing procedure follows steps described below

PKG specifies bilinear map where both groups $G_1 =< P >$ and G_2 has same prime order q. Generally $(G_1,+)$ and $(G_2,*)$. But sometimes G_1 can be defined as multiplicative group.

$$e : G_1 \times G_1 \rightarrow G_2$$

Signer: Alice selects private key randomly $r \in Z_q^*$ and publishes her public key $P_A = rP$ and computes signature $\delta = H_1(m)^r \in G_1$

Verifier: Receiving $< M, \delta >$ from Alice, Bob accept the message iff $e(P, \delta) = e(P_A, H_1(m))$

Efficiency: Signing is too easy, only one hashing and one modular exponentiation. Verification has more computation than signing. It needs two pairing calculation.
There are several signature schemes based on pairings. In this chapter we just stated on BLS signature. Hess and Cha & Cheon's signature schemes are described on further chapter.

4.6 SECURITY OF IDENTITY BASED CRYPTOGRAPHY

Even though there exist other implementation of identity based cryptography, Boneh and Franklin realization is preferred and all later researches are built on this scheme. Since Boneh and Franklin implementation uses and thus security relies on bilinear pairings, we need to check security of bilinear function. The construction of a bilinear map comes with a number of complexity implications.

Theorem 1: The Discrete Log Problem in G_1 is no harder than the Discrete Log Problem in G_2.

Proof: Consider $Q = aP$ (additive notation), though a is unknown. Solving the Discrete Log Problem involves finding a for a given P and a random Q.
Note that:

$$e(P,Q) = e(P,aP) = e(P,P)^a$$

Thus, we can reduce the Discrete Log Problem in G_1 to the Discrete Log Problem in G_2. Given P∈ G1 and a random Q∈ G1, and noting that the mapping e is easily computable, we can compute $\log_P(Q)$ as follows:

- Compute $P' = e(P,P)$
- Compute $Q' = e(P,Q)$
- Calculate $a = \log_{P'}(Q')$ in G_2
- a is also $\log_P(Q)$

Theorem 2: The Decisional Diffie-Helman (DDH) is easy in G_1.

Proof: Solving the DDH problem involves distinguishing:

Given $< P, aP, bP, cP >$ how to know if $c = ab$ and thus adversary can get significant information in deciding DDH

Calculate $x = e(aP, bP)$ and $y = (P, cP)$

$x = y$ iff $c = ab$

$x = y$

$e(aP, bP) = e(P, cP)$

$e(P, abP) = e(P, cP) \Leftrightarrow c = ab$

4.7 JOUX'S TRIPARTITE KEY EXCHANGE

Bilinear pairings not only for identity-based cryptographic schemes but used for other cryptographic schemes. One of them is Joux's Tripartie Key Exchange (Joux, 2000). He proved that key sharing among three users can be done one round while it needs number of interactions in conventional way. In this part we will discuss Diffie-Hellam key agreement and Joux's.

Assume Alice, Bob and Carl want to agree on a key. Procedure goes as follows:

Diffie-Hellman Key agreement among three users.

Given g primitive root of Z_p where p is some large prime or prime power.

Users Alice, Bob and Carl calculate the followings

Alice: selects private key $a \in Z_p$ randomly and calculates public key g^a

Bob: selects private key $b \in Z_p$ randomly and calculates public key g^b

Carl: selects private key $c \in Z_p$ randomly and calculates public key g^c

They all publish public keys. Following picture depicts it.

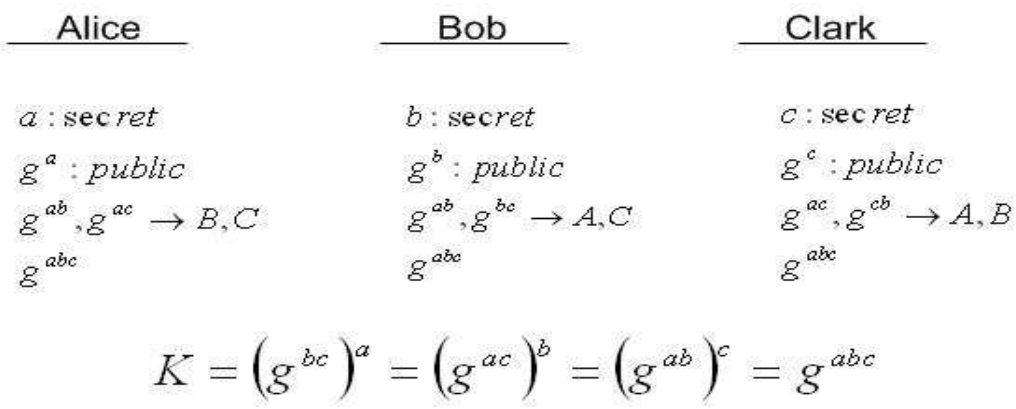

Figure 4.4 Diffie-Hellman Key agreement

Joux's Key Agreement among three users.

Each user chooses a secret key and calculates public key. All of them have private and public keys (a, aP), (b, bP) and (c, cP), private keys $a, b, c \in Z_q^*$ selected at random. And public keys $aP, bP, cP \in G_1$. We say that in one round they can agree on a key. Picture below illustrates procedure

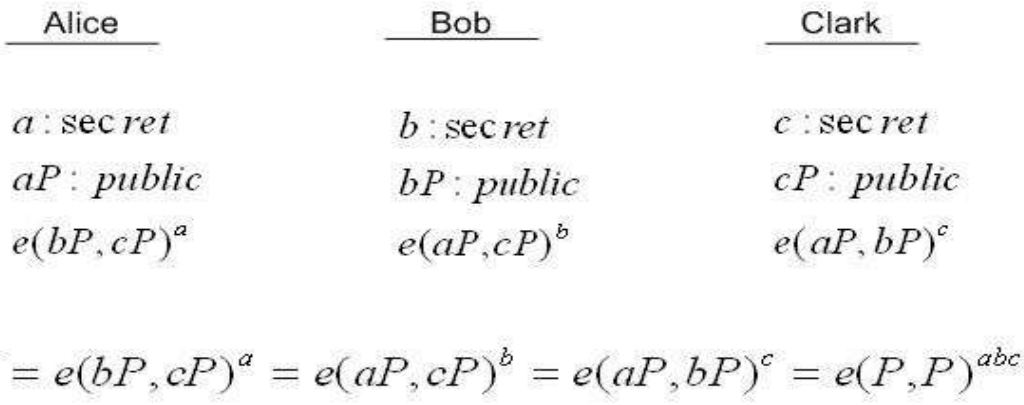

Figure 4.5 One round key agreement

4.8 OPEN PROBLEMS

As denoted in 4.7, Joux was able to build one round key agreement scheme for 3 users. What if there are lots of users need to make agreement on a key?

Bilinear function helped and used by Joux because two points are bilinear function parameters. If we have four users then Joux scheme can not make a one round key agreement. In order to make a one round key agreement for n users, then function must be $n-linear$ instead of bilinear. Since *n*-linear function has not been developed yet, this problem is still open. No practical solution has ever been realized.

One another open problem is that we define bilinear mapping as:

$$e : G_1 \times G_1 \rightarrow G_2$$

or

$$e : G_1 \times G_2 \rightarrow G_T$$

In both cases mapping is not self-mapping.

$$e : G_1 \times G_1 \rightarrow G_1$$

This can be tried out, result will probable be much more useful.

CHAPTER 5

PAIRING BASED CRYPTOGRAPHY LIBRARY

5.1 ABOUT

Pairing Based Cryptography Library (PBC) is developed at Stanford University by Ben Lynn. PBC is freely distributed C library built on GMP that performs mathematical operations. It is dedicated to practice on the new field of cryptography, pairing based cryptography that turns around some specific functions. Great thanks to Ben Lynn for his efforts to bring an opportunity to practice on pairing based cryptography.

5.2 INSTALL

Since PBC is built on GMP, we need to install GPM first. After installation of GMP we can install PBC.

5.2.1 GMP

GNU Multiple Precision (GMP) arithmetic library developed for mathematical operations without limitation except the memory of machine in which GMP runs on. The main focus on GMP is its performance regardless small and huge operands. Optimized algorithms are used with great care. Generally GMP is used in the field of cryptographic applications, algebra systems, computational algebra and etc.

Download:

As GMP is freely distributed, one can download it from http://gmplib.org/#DOWNLOAD. Move the GMP folder under /usr/src/ directory.

Install:

Using terminal go to the GMP directory

```
# cd /usr/src/gmp
```

Now, we are ready to install GMP library. Use following commands

```
#./configure
#make
```

For self-test run

```
#make check
```

Finally

```
#make install
```

For further information GMP website `http://gmplib.org/` will be useful.

5.2.1 PBC

Pairing Based Cryptography library uses GMP library for underlying mathematical operations due to its limitless precision and high performance. Since we already installed GMP, we can go on to PBC.

Download:

PBC website `http://crypto.stanford.edu/pbc/` provides great documentation not only for PBC but also tutorial about underlying mathematical topics.

Download source file from `http://crypto.stanford.edu/pbc/download.html`

Move the PBC folder to the

```
# cd /usr/src/
```

Install:

Open terminal and go to the PBC folder

```
# cd /usr/src/pbc
```

And use following commands to install PBC.

```
#./configure
#make
#make install
```

Successful installation creates libpbc.a static and libpbc.so dynamic library files in `#/usr/local/lib` directory.

5.3 SAMPLE CODE

- Interpreter pbc/pbc

Without writing any c code, we can work with pbc, so we can use it as calculator. Go to the pbc folder with terminal and type the command

```
#pbc/pbc
```

This is will start waiting for command.

Try the pbc codes below

```
a=rnd(Zr)        // a is randomly selected number modulo r
                 // defined in a.param in      param folder
a                // will print the value of a.


b=rnd(Zr)
A=rnd(G1)        // A is randomly selected point from G1
                 // elliptic curve group.
A                // will print the value of point A
B=rnd(G1)
```

Now, using bilinearity property of pairings, let's calculate pairing function. Result of two pairing functions must be identical

```
pairing(A^a, B^b)
pairing(A^b, B^a)
```

Following picture is screenshot from pbc interpreter

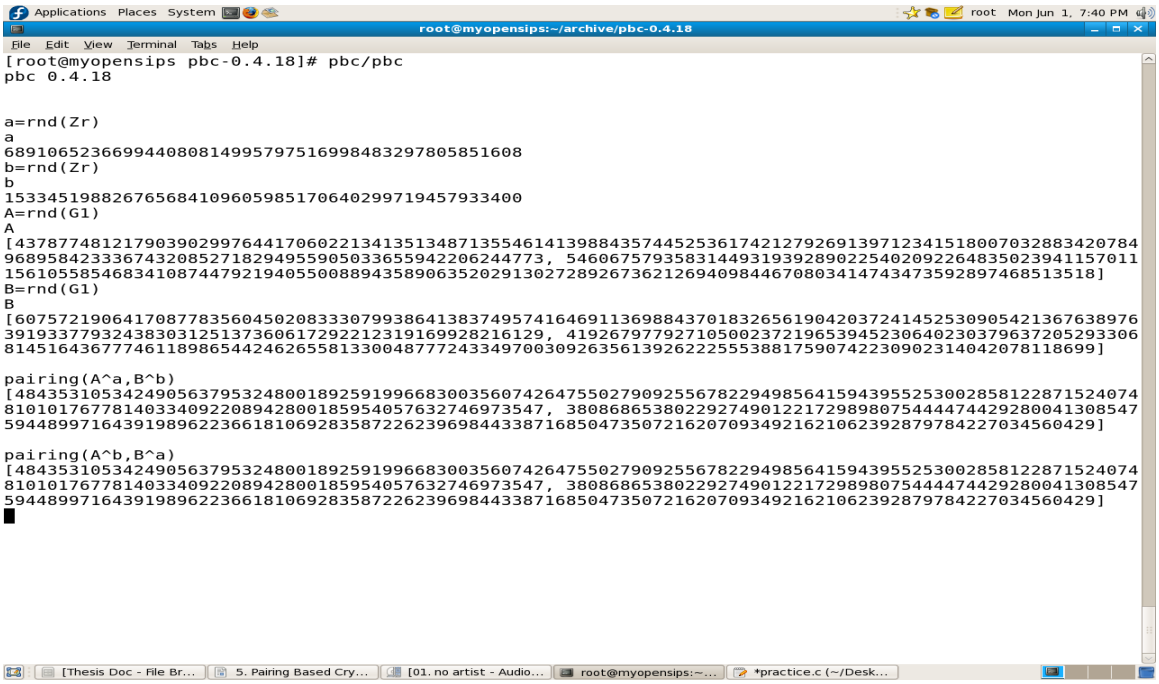

Figure 5.1 Screenshot from pbc interpreter

- Create your own .c file including pbc code

Now, let's create example.c file and and include pbc.h with giving complete path.

```
#include <stdio.h>
#include <math.h>
#define PBC_DEBUG
#include "pbc.h"
#include "gmp.h"
```

and also include a.param inside param folder.

```
static char *aparam =
"type a\n"
"q
  87807107996633125224377819847540498158068831994142082110286533
  99266
 47563088022295707862517942266222142315585876958231745927771336731748132492512999822479 1\n"
"h
  12016012264891146079388821366740534204802954401251311822919615
  131047
  20728935970453110284480218390653778677 6\n"
"r 730750818665451621361119245571504901405976559617\n"
"exp2 159\n"
"exp1 107\n"
"sign1 1\n"
"sign0 1\n";
```

```
        char *input = aparm;
        pairing_t pairing;

        pairing_init_inp_buf(pairing, input, strlen(input));   //
initialize pairing paramaters

        element_t a, b, A, A1, B, B1, pair1, pair2;

        // initialize types of elements
        element_init_Zr(a, pairing);
        element_init_Zr(b pairing);
        element_init_G1(A, pairing);
        element_init_G1(B, pairing);
        element_init_G1(A1, pairing);
        element_init_G1(B1, pairing);
        element_init_GT(pair1, pairing);
        element_init_GT(pair2, pairing);

        // initialize all elements.
        element_random(a);           // select random number from Z_r
        element_printf("number a = %B\n", a);  // print out value of a
        element_random(b);
        element_random(A);              // selec random point from G1
        element_printf("point A = %B\n", A);    // print out point A
        element_random(B);
```

Now let's apply pairing function and compare them. But first we need to calculate aA and bB

```
        element_mul_zn(A1, A, a);    // point exponentiation,  A1=aA
        element_mul_zn(B1, B, b);
        pairing_apply(pair1, A1, B1, pairing);   // pairing function
                                                 // pair1=e(A1,B1)

        element_mul_zn(A1, A, b);
        element_mul_zn(B1, B, a);

        pairing_apply(pair2, A1, B1, pairing);

        if (!element_cmp(pair1, pair2))     // comapare pair1 and pair2
```

```
            printf("pairs are equal\n");
    else
            printf("pairs are not equal\n");
```

- How to compile

While compiling we have to give link to libpbc.a or libpbc.so to understand the pbc codes.

```
# gcc –Wall example.c –L. –lpbc –lm –o example
```

Wall : warn and give all errors.

L. : address of library folder

lpbc : short form of libpbc

lm : short form of lmath library

o example : declare *example* as a name of executable file

We keep pbc sample code very short. Extended information can be found http://crypto.stanford.edu/pbc/manual/ch05.html.

CHAPTER 6

TEST ENVIRONMENT

6.1 OPENSIPS

6.1.1 Overview

OpenSIPS (Open SIP Server) is an open source SIP server program developed by Voice System group back in 2005. It includes more other features like application-level functionalities. OpenSIP is continuation of OpenSER (Open SIP Express Router) project. Using C language it is developed and used as VOIP registrar, location server, proxy server, redirector server and followings

- SIP presence agent
- SIP IM server (chat and end-2-end IM)
- SIP to SMS gateway (bidirectional)
- SIP to XMPP gateway for presence and IM (bidirectional)
- SIP load-balancer or dispatcher
- SIP front end for gateways/asterisk
- SIP NAT traversal unit
- SIP application server

6.1.2 Installation

OpenSIPS packages are distributed on many websites, but better to download from it's own site

```
http://opensips.org/pub/opensips/latest/src/
```

After downloading, extract if it is zipped, move OpenSIPS folder to the /usr/src directory. Before installing OpenSIPS, there need to be done some other installations.

1. Download and install the mysql and mysql-devel packages from internet.

```
# yum install mysql mysql-server
# yum install mysql mysql-devel
```

or from Package Manager of Fedora 8, used through our project implementation, mysql and mysql-devel can be installed.

2. Open Terminal and go to the /usr/src/opensips directory and do following commands to install OpenSIPS.

```
# ./configure
# make
# make install
```

3. To create the MySQL database, you have to use the opensips_mysql.sh script.

```
# /usr/sbin/ opensips_mysql.sh create
```

After the installation of MySQL database the default passwords are;

```
username : admin@my opensips.org
pass    : opensipsrw
```

6.1.3 How to run

Before starting OpenSIPS, check if mysql is running. If mysql is running then we are ready to start OpenSIPS. Use following commands in terminal

```
# /usr/sbin/
# ./opensips
```

File System Organizations

/usr/src/	:	source files
/usr/sbin/	:	scripts (OpenSIPS, OpenSIPSctl, OpenSIPS_mysql.sh)
/etc/init.d	:	OpenSIPS (start|stop| restart)

/usr/local/etc/opensips/opensips.cfg: general configurations and configuring authentication

/usr/local/sbin : executable

Links

Official site: http://www.opesips.org/

Download: http://www.opensips.org/Resources/Downloads

Installation: http://www.opensips.org/Resources/Install

Information: http://www.voip-info.org/wiki/view/opensips

Documentation: http://www.opensips.org/Resources/Documentation

6.1.4 Embedding IBS code

Challenge/Response technique is standardized in SIP protocol and used in all SIP servers and clients as in OpenSIPS and PJSIP. OpenSIPS checks response message from client and authenticates if true. In standard MD5 algorithm is used to calculate response. Now, we are replacing it with IBS method by Hess and Cha&Cheon.

To change it, go to the file `api.c`

```
#/usr/src/opensips-1.4.4-notls/modules/auth/api.c
```

On the apic.c change the related function.

6.1.5 Recompile

Now, after adding new IBS code, OpenSIPS must be recompiled. To do this, go to the opensips folder

```
#/usr/src/opensips-1.4.4-notls/
```

and run to compile it.

```
#make
#make install
```

Here it gives an error, because we will link pbc library manually. Run the following

commands to compile it completely and correctly.

```
#export SIP_DOMAIN=”myopensips.org”
#export LD_LIBRARY_PATH=/usr/local/lib


1- /usr/src/opensips-1.4.4-notls/modules/auth_db/
gcc -shared  -Wl,-O2 -Wl,-E    authdb_mod.o authorize.o   -o
auth_db.so -L /usr/local/lib -Wl,-rpath /usr/local/lib  -l pbc

2- /usr/src/opensips-1.4.4-notls/modules/auth/
gcc -shared  -Wl,-O2 -Wl,-E    auth_mod.o api.o challenge.o common.o
index.o nonce.o rfc2617.o rpid.o -o auth.so -L /usr/local/lib -Wl,-
rpath /usr/local/lib -l pbc
```

6.2 PJSIP

6.2.1 About

PJSIP is an open source SIP stack, has been actively developed since 2003. It has history before 2003. But evaluation of SIP protocol brings changes on PJSIP. Currently 3rd generation is used. In this thesis PJSIP is used to create SIP clients and authenticated to SIP registrar, typically OpenSIPS, in order to benchmark authentication algorithms. Throughout this thesis, we used source pjsip package developed in C for Unix-like systems.

PJSIP provides clear and user-friendly documentation that is strongly important to be used by new users. We can list some important features of PJSIP as follows

- Portability: PJSIP is multiplatform. Once you write an application is enough to run on all (Windows, Linux, Unix-like systems, Windows Mobile, MacOS, Sybmain OS and etc.)
- Very small footprint: With very small size. Plays important role in embedding devices where space cost is important.
- High Performance
- Other features: SIP user agent, IM, call transfer multi user registration and etc.

For more information manual documentations can be found from PJSIP website http://www.pjsip.org/

6.2.2 Installation

Download:

Download the pjsip source *.zip* or *.tar.bz2* from the link http://www.pjsip.org/download.htm. If you are going to use on Linux platform then download *.tar.bz2* file, if Windows then *.zip* file.

Install:

After downloading, move the unzipped file to the `#/usr/src` directory. And prior to build pjsip, `config_site.h` file must be created `pjlib/include/pj/config_site.h (it can just be an empty file).`

Using Terminal, go to the directory of pjsip `#/usr/src/pjsip/,` and run following commands

```
#./configure
#make dep
#make
```

If command above are executed without error means pjsip is ready to use.

Note: After successful installation, executable file can be found in corresponding subdirectory.

Uninstall:

Once successfully install pjsip, it can be used as much as u need. In case whenever you change the source code, it must uninstalled and rebuilt. To uninstall use following command

```
#make realclean
```

Remove all generated files (object, libraries, binaries, and dependency files)

6.2.3 How to run

Run:

To run pjsip go to the following directory and use commands.

```
#/usr/src/pjsip/pjsip-apps/bin
```

and run the executable file under bin directory.

```
#./pjsua-i686-pc-linux-gnu
```

Following picture show command-line user-interface

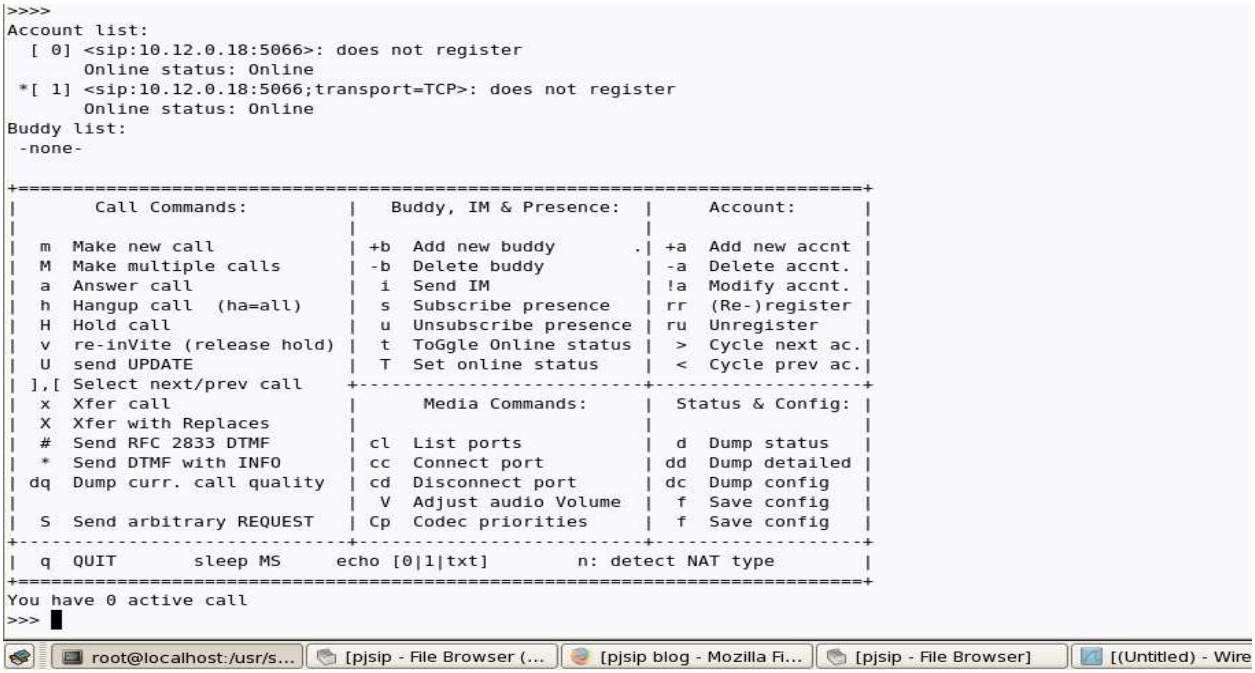

Figure 6.1 PJSIP command-line user-interface

Stop:

Command-line user-interface provides great help. We can stop just by typing *q*.

```
#q
```

6.2.4 Embedding IBS code

Since the main purpose of this project is to increase the security of SIP authentication even though authentication time increases, we changed the algorithm used in authentication. As it is known MD5 is a unique technique that has been used in SIP authentication for a long time. In this section I will briefly state how to embed code and MD5 is replaced with IBS scheme.

Creating digest response using MD5 algorithm is in the file `/pjsip/src/pjsip/sip_auth_clinet.c`

From that file comment or delete the function named `pj_status_t respond_digest(…)` and put IBS code there.

In order to run PJSIP properly, it needs to be rebuilt. And to understand the pbc (pairing based crypto) codes, `libpbc.a` or `libpbc.so` must be linked before running.

6.2.5 Recompile

Just by embedding the code it will not work. PJSIP must be recompiled. Fist of all delete all created object, executable and etc files typing the command

```
#make realclean.
```

Now, we can build pjsip as in installation.

```
#./configure
#make dep
#make
```

Here, it will give some error related with pbc codes. Because pbc library is not includes while installation. That's why make dep will give some undefined errors.
Object and library files those are include pbc codes will not be created. Solution to this problem is that we can execute gcc command including link to the pbc library.
Here are some errors I met during compilation and fixed them with followings.

```
1- export LD_LIBRARY_PATH = /usr/local/lib

2- /pjsip/build/

gcc -c -Wall  -DPJ_AUTOCONF=1 -O2    -I../include -
I../../pjlib/include -I../../pjlib-util/include -I../../pjnath/include
-I../../pjmedia/include -o output/pjsip-i686-pc-linux-
gnu/sip_auth_client.o ../src/pjsip/sip_auth_client.c -L/usr/local/lib
-lpbc -lgmp


3- /pjsip/build/

gcc -o ../bin/pjsip-test-i686-pc-linux-gnu output/pjsip-test-i686-pc-
linux-gnu/main.o  output/pjsip-test-i686-pc-linux-gnu/dlg_core_test.o
output/pjsip-test-i686-pc-linux-gnu/dns_test.o  output/pjsip-test-
i686-pc-linux-gnu/msg_err_test.o  output/pjsip-test-i686-pc-linux-
gnu/msg_logger.o  output/pjsip-test-i686-pc-linux-gnu/msg_test.o
output/pjsip-test-i686-pc-linux-gnu/regc_test.o  output/pjsip-test-
i686-pc-linux-gnu/test.o  output/pjsip-test-i686-pc-linux-
gnu/transport_loop_test.o  output/pjsip-test-i686-pc-linux-
gnu/transport_tcp_test.o  output/pjsip-test-i686-pc-linux-
gnu/transport_test.o  output/pjsip-test-i686-pc-linux-
gnu/transport_udp_test.o  output/pjsip-test-i686-pc-linux-
gnu/tsx_basic_test.o  output/pjsip-test-i686-pc-linux-gnu/tsx_bench.o
output/pjsip-test-i686-pc-linux-gnu/tsx_uac_test.o  output/pjsip-test-
i686-pc-linux-gnu/tsx_uas_test.o  output/pjsip-test-i686-pc-linux-
gnu/txdata_test.o  output/pjsip-test-i686-pc-linux-gnu/uri_test.o
output/pjsip-test-i686-pc-linux-gnu/inv_offer_answer_test.o -
L/usr/src/pjproject-1.0.1/pjlib/lib -L/usr/src/pjproject-1.0.1/pjlib-
util/lib -L/usr/src/pjproject-1.0.1/pjnath/lib -L/usr/src/pjproject-
1.0.1/pjmedia/lib -L/usr/src/pjproject-1.0.1/pjsip/lib -
L/usr/src/pjproject-1.0.1/third_party/lib  -lpjsua-i686-pc-linux-gnu -
lpjsip-ua-i686-pc-linux-gnu -lpjsip-simple-i686-pc-linux-gnu -lpjsip-
i686-pc-linux-gnu -lpjmedia-codec-i686-pc-linux-gnu -lpjmedia-i686-pc-
linux-gnu -lpjnath-i686-pc-linux-gnu -lpjlib-util-i686-pc-linux-gnu -
lresample-i686-pc-linux-gnu -lmilenage-i686-pc-linux-gnu -lsrtp-i686-
```

```
pc-linux-gnu -lgsmcodec-i686-pc-linux-gnu -lspeex-i686-pc-linux-gnu -
lilbccodec-i686-pc-linux-gnu -lportaudio-i686-pc-linux-gnu -lpj-i686-
pc-linux-gnu -lm -luuid -lnsl -lrt -lpthread  -lasound -lssl -lcrypto
-L/usr/local/lib -lpbc -lgmp
```

4- /pjsip-apps/build/

```
gcc -c -Wall  -DPJ_AUTOCONF=1 -O2    -I../../pjsip/include -
I../../pjlib/include -I../../pjlib-util/include -I../../pjnath/include
-I../../pjmedia/include -o output/pjsua-i686-pc-linux-gnu/pjsua_app.o
../src/pjsua/pjsua_app.c -L /usr/local/lib -lpbc -lgmp
```

5- /pjsip-apps/build/

```
gcc -o ../bin/pjsua-i686-pc-linux-gnu output/pjsua-i686-pc-linux-
gnu/main.o  output/pjsua-i686-pc-linux-gnu/pjsua_app.o -
L/usr/src/pjproject-1.0.1/pjlib/lib -L/usr/src/pjproject-1.0.1/pjlib-
util/lib -L/usr/src/pjproject-1.0.1/pjnath/lib -L/usr/src/pjproject-
1.0.1/pjmedia/lib -L/usr/src/pjproject-1.0.1/pjsip/lib -
L/usr/src/pjproject-1.0.1/third_party/lib  -lpjsua-i686-pc-linux-gnu -
lpjsip-ua-i686-pc-linux-gnu -lpjsip-simple-i686-pc-linux-gnu -lpjsip-
i686-pc-linux-gnu -lpjmedia-codec-i686-pc-linux-gnu -lpjmedia-i686-pc-
linux-gnu -lpjnath-i686-pc-linux-gnu -lpjlib-util-i686-pc-linux-gnu -
lresample-i686-pc-linux-gnu -lmilenage-i686-pc-linux-gnu -lsrtp-i686-
pc-linux-gnu -lgsmcodec-i686-pc-linux-gnu -lspeex-i686-pc-linux-gnu -
lilbccodec-i686-pc-linux-gnu -lportaudio-i686-pc-linux-gnu -lpj-i686-
pc-linux-gnu -lm -luuid -lnsl -lrt -lpthread  -lasound -lssl -lcrypto
-L/usr/local/lib -lpbc -lgmp
```

6- Send sip msg just once while registering.

```
file :  sip_config.h
"#define PJSIP_T1_TIMEOUT    500" -->  "#define PJSIP_T1_TIMEOUT
      100000"
```

7- /pjsip-apps/build/

```
gcc -o ../bin/pjsua-i686-pc-linux-gnu \
             output/pjsua-i686-pc-linux-gnu/main.o  output/pjsua-i686-
pc-linux-gnu/pjsua_app.o -L/usr/src/pjproject-1.0.1/pjlib/lib -
L/usr/src/pjproject-1.0.1/pjlib-util/lib -L/usr/src/pjproject-
1.0.1/pjnath/lib -L/usr/src/pjproject-1.0.1/pjmedia/lib -
L/usr/src/pjproject-1.0.1/pjsip/lib -L/usr/src/pjproject-
1.0.1/third_party/lib  -lpjsua-i686-pc-linux-gnu -lpjsip-ua-i686-pc-
linux-gnu -lpjsip-simple-i686-pc-linux-gnu -lpjsip-i686-pc-linux-gnu -
lpjmedia-codec-i686-pc-linux-gnu -lpjmedia-i686-pc-linux-gnu -lpjnath-
i686-pc-linux-gnu -lpjlib-util-i686-pc-linux-gnu -lresample-i686-pc-
linux-gnu -lmilenage-i686-pc-linux-gnu -lsrtp-i686-pc-linux-gnu -
lgsmcodec-i686-pc-linux-gnu -lspeex-i686-pc-linux-gnu -lilbccodec-
i686-pc-linux-gnu -lportaudio-i686-pc-linux-gnu -lpj-i686-pc-linux-gnu
-lm -luuid -lnsl -lrt -lpthread  -lasound -lssl -lcrypto -
L/usr/local/lib -lpbc -lgmp
```

CHAPTER 7

PERFORMANCE EVALUATION

7.1 PURPOSE

Implementation and result of identity based signature schemes on SIP authentication comes in this chapter. Conventional SIP authentication uses MD5, one-way hash function. More information and MD5 scheme will be discussed in 7.2. So, shortly saying that MD5 algorithm is no more secure against some attacks like dictionary attack in case of weak passwords.

Because of such disadvantages, we replaced the MD5 algorithm with identity based signature scheme. Here, we implemented Hess's and Cha&Cheon's identity based signature algorithms.

Prior to implementing the IBS algorithms, let's see SIP Digest authentication in detail.

7.2 CONVENTIONAL SIP AUTHENTICATION

SIP networks are subject to various threats and attacks. Snooping, modification, spoofing, reply and denial of service attacks are mentioned in the literature (Chapter 2). SIP security mechanisms can be divided into two categories, hop-by-hop and end-to-end protection mechanism as shown in Figure 7.1. The hop-by-hop security mechanisms are transport level security mechanism such as TLS and network level security mechanism such as IPSec (Eastlake, 2009) and (Kent et al., 2005). End-to-end protection mechanisms are S/MIME and the HTTP Digest authentication method.

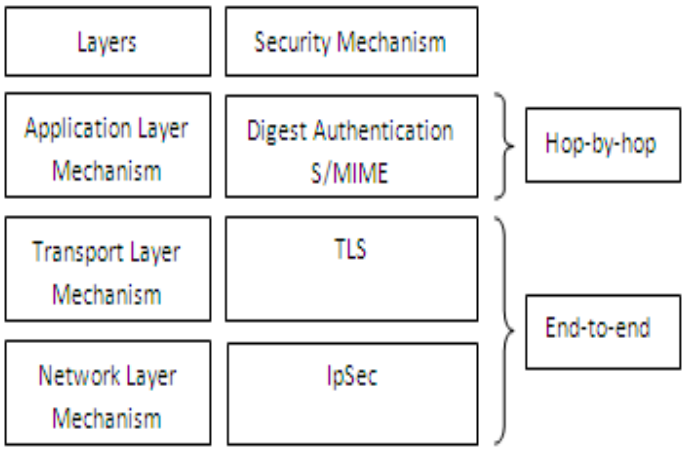

Figure 7.1 SIP Security Mechanism

The HTTP Digest authentication is based on a challenge-response scheme and is the common authentication method for most SIP proxy server applications. The message flow of the HTTP Digest Authentication can be seen in Figure 7.2.

After the clients Request message the server challenges the client with a nonce asking to calculate a function that involves the mutual secret key and the nonce value. If the outcome calculated on the client side matches the server side calculation, the client will be authenticated.

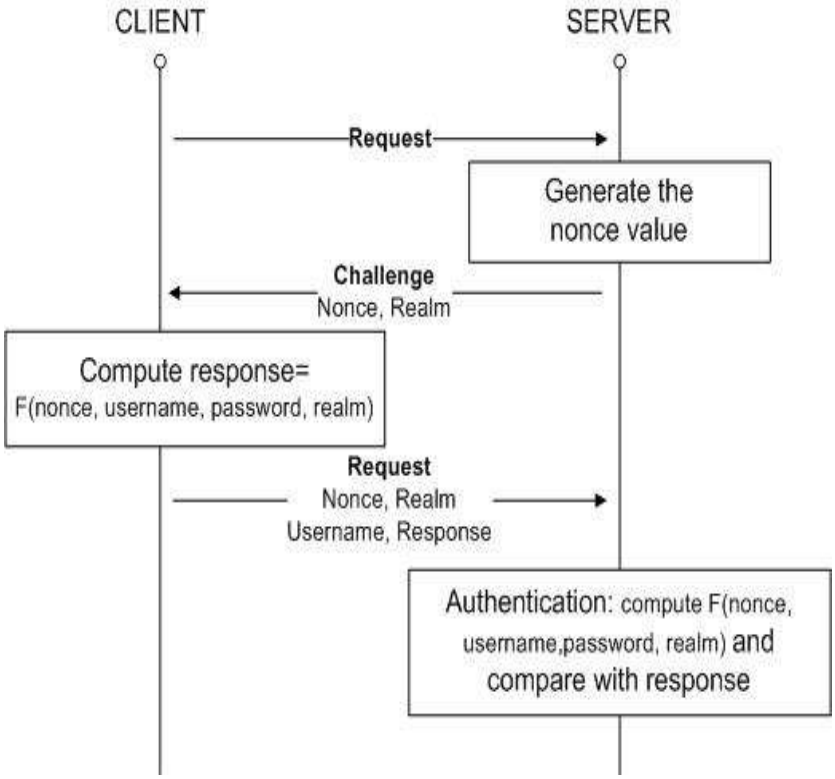

Figure 7.2 The HTTP Digest Authentication method

Digest Authentication does not provide confidentiality protection except password protection. The rest of the request and response can be obtained by the eavesdropper. Digest Authentication offers only limited integrity protection for the messages in both directions. Most header fields and their values can be changed with a man-in-the-middle attack. Many needs of a secure HTTP transaction can not met by the HTTP Digest Authentication. Digest authentication is not used for a process that requires privacy protection (Rosenberg et al., 2002).

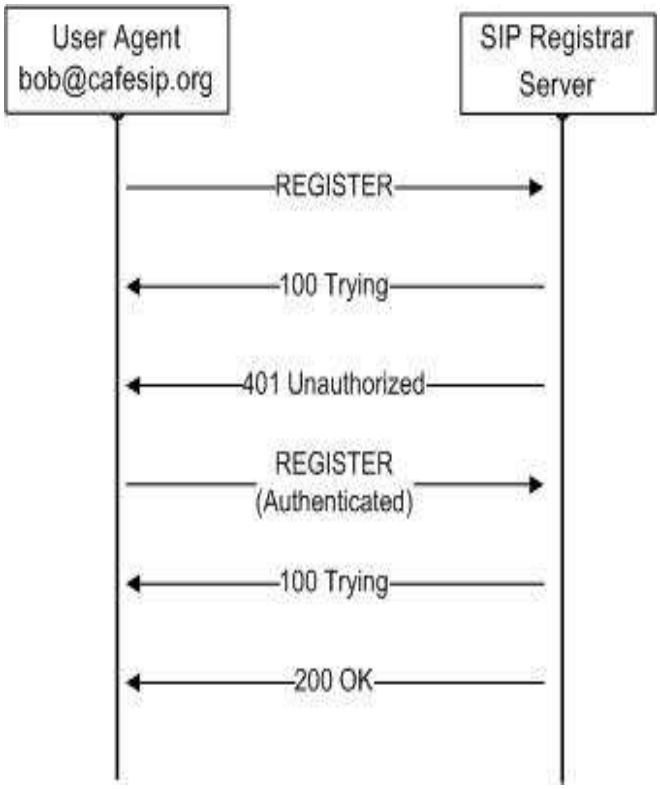

Figure 7.3 SIP Registration Procedure

The SIP message exchange of the registration procedure is shown in Figure 7.3. Here the HTTP digest authentication is used. The “401 Unauthorized” message includes the initial nonce value. The outcome of the function involving the secret key and the nonce is included in the second REGISTER message.

7.3 HESS IBS AUTHENTICATION

We will use the following scheme in (Hess, 2002) to sign our nonce value received from the server in our SIP registration procedure. Having $(G,+)$ and $(G_T,\cdot)$ denote cyclic groups of prime order l, P element of G, P a generator of G and $e:G \times G \rightarrow G_T$ be a pairing which satisfies the bilinear and non-degenerate properties defined in (Hess, 2002). Furthermore we define the hash functions

$h: \{0,1\}^* \times G_T \rightarrow (Z/lZ)^\times$, $H: \{0,1\}^* \rightarrow G^*$

where $G^* := G \setminus \{0\}$. The ID based signature scheme consists of 4 algorithms, **Setup**, **Extract**, **Sign** and **Verify**. The entities involved are the trusted authority (TA), the signer and the verifier.

Setup: TA selects a random integer t element of $(Z/lZ)^\times$, computes $P_{pub} = t\,P$. t remains secret. TA publishes P_{pub}.

Extract : The signers request private keys $SA = tH($IDA), IDA is the signer A's identity.

Sign: To sign message m the signer selects arbitrary K element of G and a random integer r element of $(Z/lZ)^\times$, and computes

$q = e(K,P)^r$

$V = h(m,q)$

$U = V * SA + K * r$

Verify: The verifier receives message m and the signature (U,V) and computes

$q = e(U,P)\, e(H(\text{IDA}), -P_{pub})^V$

Accepts if and only if $V = h(m,q)$

In Figure 7.4 the computation and message exchange in Hess' algorithm is shown.

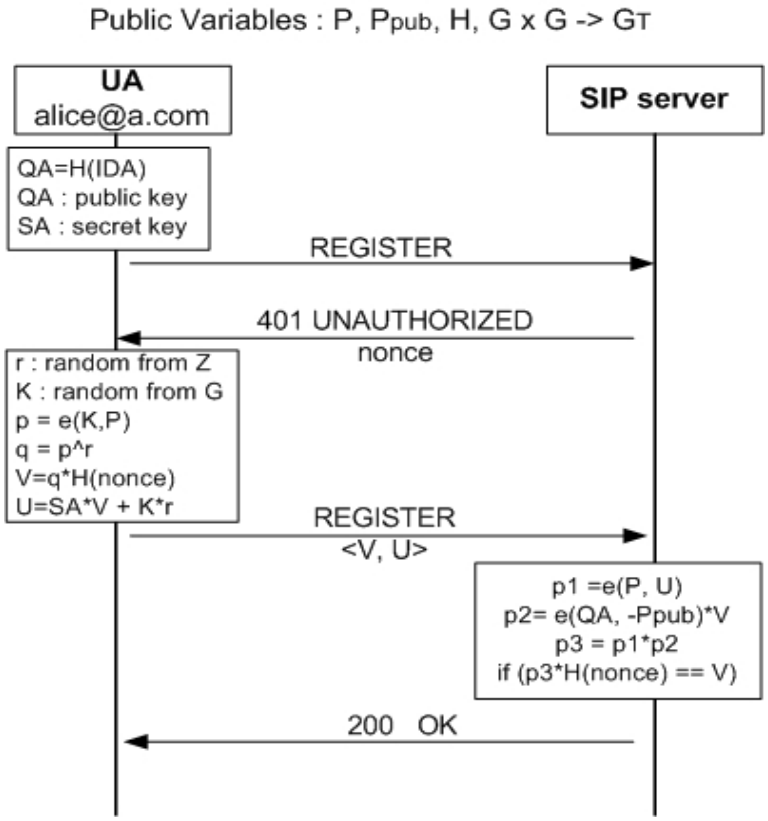

Figure 7.4 Hess's ID Based Signature Algorithm

Figure 7.5 shows the second REGISTER message body within the SIP registration procedure which transfers the signature (*U,V*) in its "response" part.

```
REGISTER sip:10.0.2.56 SIP/2.0
Via: SIP/2.0/UDP 10.0.3.196:5066;rport;branch=z9hG4bKPja23bba07-bc8b-48da-
8836-7ec273d17cff
Max-Forwards: 70
From: <sip:1019@10.0.2.56>;tag=d0de1799-06e3-4b7d-bb5e-651375a346dc
To: <sip:1019@10.0.2.56>
Call-ID: 2c2fcb40-77c9-48e1-a7bc-14dbda7ee9fd
CSeq: 35087 REGISTER
User-Agent: PJSUA v1.0.1/i686-pc-linux-gnu
Contact: <sip:1019@10.0.3.196:5066>
Expires: 5
Authorization: Digest username="1019", realm="10.0.2.56",
nonce="49cf9670000000132dcba7d98ce4999f6d250423cbd1cd20",
uri="sip:10.0.2.56",
response="610577919698523737380492419259910740701680651375*[6927747816
372485968266333607227404719049577254987149535925386990567101873435940
462106349911186711062894089727404374106005015317244822352883880212818
714011,
153266535503486388326928735901626300236960356714707957967639643329794
866664068805777809046417161994711380936690623466925794050734755580418
9000260811475016]
--end msg--
```

Figure 7.5 Register message with signature (*U,V*) in the response in Hess's algorithm

7.4 CHA AND CHEON IBS ALGORITHM

Similar to Hess's algorithm having $(G,+)$ denote cyclic groups of prime order l with generator P and $e:G \times G \rightarrow G_T$ be a pairing which satisfies the bilinear and non-degenerate properties . Furthermore we define the hash functions

$H_1: \{0,1\}^* \rightarrow G^*$, $H_2: \{0,1\}^* \times G \rightarrow (Z/lZ)^\times$

The ID based signature scheme consists of 4 algorithms, **Setup**, **Extract**, **Sign** and **Verify**. The entities involved are the trusted authority (TA), the signer and the verifier.

Setup: TA selects a random integer s element of $(Z/lZ)^\times$, computes $P_{pub} = s\ P$. s remains secret. TA publishes P_{pub}.

Extract : The signers request private keys for $QA = H(\text{IDA})$, $SA = s\ QA$, IDA is the signer A's identity.

Sign: To sign message m the signer selects a random integer r element of $(Z/lZ)^\times$, and computes

$U = r * QA$

$V = (r+H(\text{nounce},U))\ SA$

Send (U,V)

Verify: The verifier receives message m and the signature (U,V) and computes

Accepts if and only if

$e(P,V) = e(P_{pub}, U + H(\text{nounce},U)\ QA)$

In Figure 7.6 the computation and message exchange in Cha and Cheon's algorithm is shown.

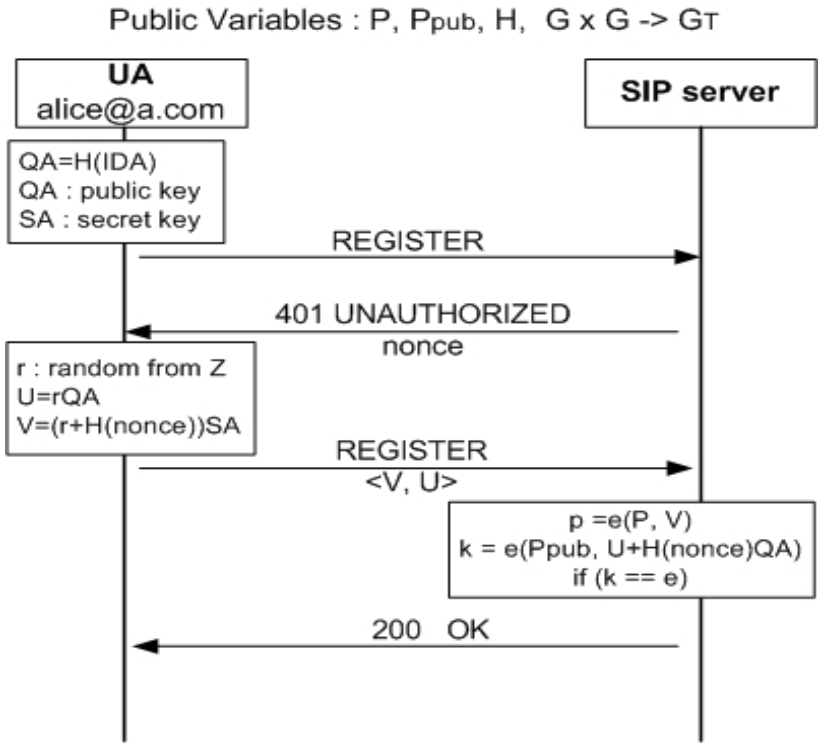

Figure 7.6 Cha and Cheon's ID Based Signature Algorithm

REGISTER sip:10.0.2.56 SIP/2.0
Via: SIP/2.0/UDP 10.0.3.196:5066;rport;branch=z9hG4bKPj4269e061-67b4-4878-ae9f-97e89ae2665e
Max-Forwards: 70
From: <sip:1020@10.0.2.56>;tag=f2c7d45a-422a-40df-b3e2-ad9cc72e3a08
To: <sip:1020@10.0.2.56>
Call-ID: fb8fa94b-e6b8-4b11-9775-1465061bbd57
CSeq: 49325 REGISTER
User-Agent: PJSUA v1.0.1/i686-pc-linux-gnu
Contact: <sip:1020@10.0.3.196:5066>
Expires: 5
Authorization: Digest username="1020", realm="10.0.2.56",
nonce="49cf6d1300000538481c783d85823fc13a1889850f513d25",
uri="sip:10.0.2.56",
response="[77985158657001098361764494998847159577296566042739141686830
22510495758053327034427586222439552128541460598390747088722232015676879620536507590966066903317,
48172749197828759309449171273136474945955071275868351084548064642457639101551845357892120806584381630108853934596667122469874661494722437171746668297558931]*[56988188066272171616826357228869414433141521648807474904836421017932413225529682033661331941987894374491098190709600161026463514793177099448101099073314904357579312820553810390649526044528337176390557351615814791760396610443587919254000717529617491943880882948575827805727873698655858219907285222347171193]
--end msg--

Figure 7.7 Register message containing the signature (*U,V*) in the response part in Cha and Cheon's algorithm

Comparing Figure 7.5 and Figure 7.7 we can see that Cha and Cheon's algorithm transfers two elliptic curve points creating a longer signature, where Hess's algorithm transfers one elliptic curve point and a number.

7.5 TESTING ENVIRONMENT

A. HARDWARE COMPONENTS

We installed the open source OpenSIPS proxy server on a PC with the following configuration:

Intel Pentium Dual CPU E2200 @ 2.20GHz,

2048 MB of RAM,

10/100 Mbps Ethernet interface cards (NICs).

The operating system is Fedora Core 6.

The PC we run the client software's has the following configuration:

Intel Pentium 4 CPU 3.20GHz,

1024 MB of RAM,

10/100 Mbps Ethernet interface card (NICs).

The operating system is Fedora Core 8.

We used a Planet switch to form a network with the PCs. Although Planet switches have some layer 2 properties, in our scenario, we used it as a hub.

B. SOFTWARE COMPONENTS

We developed a client application based on pjsip, a high performance open source SIP stack (Ismangil et al., 2003). Our client software is able to send REGISTER and INVITE request messages to the various proxy servers and follow up the message exchange until a 200 OK message is received.

Wireshark is a very useful tool to analyze the network packets. It is a popular free network protocol analyzer (packet sniffer) having support for various protocols to display and analyze network traffic.

7.6 RESULT

Table I shows the results of our performance measurements. The timings are the average of outcomes of five different timing sets each of which are made of 20 consecutive registration runs. The first column shows the bare signing times independent of the registration procedure. The second column shows the bare verification time independent of the registration procedure. The third column is the time for the whole registration procedure shown in Figure 7.4 and Figure 7.6.

Table 7.1 Registration Timing

Authentication Algorithms	**Signing (ms)**	**Verification (ms)**	**Registration Time (sec)**
RegisterWithout Authentication	0	0	0.01127
HTTP Digest Authentication	0.005	0.005	0.03808
Cha and Cheon	16.374	36	0.78482
Hess	61.953	16	0.53603

From Table I we can see that the Digest Authentication is 20 times faster than the Cha and Cheon algorithms implementation, and 14 times faster than the Hess algorithms implementation. The Hess algorithm performs better than the Cha and Cheon's algorithm. One of the reasons is that Hess's algorithm has a lower overhead on the server side causing the serer to reply faster to consecutive registration requests.

CHAPTER 8

CONCLUSION

As mentioned in previous chapters, conventional SIP uses HTTP Digest Authentication as its standard authentication scheme and its security depends on one-way MD5 hash function. Due to the weakness of HTTP Digest Authentication against some attacks, especially dictionary attack being effective in case weak password, this authentication scheme needed to be either replaced with some other mechanisms completely or default scheme could be reconstructed in order to increase security. However, security is not the only concern but performance has also important role on SIP environment to provide users with applicable services. Even though MD5 doesn't provide strong security, performance issue makes this scheme standardized in security of SIP authentication and well accepted by SIP comprised applications. Taking into consideration about these two core issues, security and performance, we implemented two IBS schemes are Hess's and Cha & Cheon's identity-based signature schemes.

Security aspect of SIP authentication with IBS scheme is no more under threat. Because no password is used, as it brings problem of weak password guessing. One-way MD5 hash function has no place in IBS scheme. Instead much stronger hash function, maps any string to a point. Security of identity based cryptography depends on BDH assumption. To say with other words, no algorithm exists yet that can break it on polynomial time.

From our performance results we realize that approaching the HTTP Digest Authentication performance would be very difficult for a lot of different authentication mechanisms. But we can't stop asking ourselves how efficient the PBC library was implemented. A more efficient implementation might bring us to closer range.

Also the long signatures added to the SIP messages could become an important overhead in a congested network.

On the other hand if ID based signature algorithms can be further optimized they could become preferable with the advantages they provide. Being resistant to attacks like password guessing, spoofing and solving the public key management and certificate access issues could make ID based cryptosystems preferable.

The current nonce based registration procedure spends two roundtrip times. A nonce is quite useful to check the freshness of messages. No research has been made to reduce the roundtrip times. Reducing the double roundtrip times to one would increase the performance of the SIP authentication mechanisms. This could be next hot research area of SIP authentication using identity based signcryption scheme in order to sing and encrypt the message simultaneously.

REFERENCES

- A. Boldyreva, “Efficient Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffe-Hellman-group Signature Scheme”, Public Key Cryptography - Proceedings of PKC 2003, LNCS 2567, pages 31-46, 2003.

- Dan. Boneh and M. Franklin, “Identity-Based Encryption from the Weil Pairing”, Proceedings of CRYPTO 2001, LNCS 2139, pages 213-229, Springer-Verlag, 2001b.

- Dan Boneh , “The Decision Diffie-Hellman Problem”, Third Algorithmic Number Theory Symposium, 1998a.

- D. Boneh, B. Lynn, and H. Shacham, “Short Signatures from the Weil Pairing”, Advances in Cryptology - Proceedings of ASIACRYPT 2001, LNCS 2248, pages 566-582, Springer-Verlag, 2001.

- Eun-Chul Cha, Hyoung-Kee Choi and Sung-Jae Cho, “Evaluation of Security Protocols for the Session Initiation Protocol”, ICCCN 2007, Proceedings of 16th International Conference, p.611 – 616, 2007.

- J. Cha and J. Cheon, “An Identity-Based Signature from Diffie-Hellman Groups”, Public Key Cryptography -Proceedings of PKC 2003, LNCS 2567, pages 18-30, 2003.

- C. Cocks, “An Identity Based Encryption Scheme Based on Quadratic Residues”, Cryptography and Coding - Institute of Mathematics and its Applications International Conference on Cryptography and Coding -Proceedings of IMA 2001, LNCS 2260, pages 360-363, Springer-Verlag, 2001.

- Mark Collier, “Basic Vulnerability Issues for SIP Security”, CTO, SecureLogix Corporation, March 2005.

- Donald Eastlake 3rd, *Transport Layer Security Extensions, 2009,* http://www.ietf.org/internet-drafts/draft-ietf-tls-rfc4366-bis-04.txt

- Kirsten Eisentrager, Kristin Lauter, and Peter L. Montgomery, “Fast Elliptic Curve Arithmetic and Improved Weil Pairing”, Evaluation, 2003.

- R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T.Berners-Lee, *Hypertext Transfer Protocol – HTTP/1.1, 1999* , http://www.ietf.org/rfc/rfc2616.txt

- J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen and L. Stewart, *HTTP Authentication: Basic and Digest Access Authentication*, IETF, RFC 2617, 1999.

- D. Geneiatakis, A. Dagiouklas, G. Kambourakis, C. Lambrinoudakis, S. Gritzalis, S. Ehlert, D. Sisalem, "Survey of Security Vulnerabilities in Session Initiation Protocol", IEEE Communications Surveys and Tutorials, vol. 8 (3), pp. 68–81, IEEE Press, 2006.

- Kyusuk Han, Chanyeob Yeun and and Kwangjo Kim, "Design of secure VoIP using ID-Based Cryptosystem", Proc. Of SCIS 2008, Jan. 22-25, 2008.

- M. Handley and V. Jacobson, *SDP: Session Description Protocol, 1998,* http://www.ietf.org/rfc/rfc2327.txt

- F. Hess, "Efficient Identity Based Signature Schemes Based on Pairings", Selected Areas in Cryptography - Proceedings of SAC 2002, LNCS 2595, pages 310-324, Springer-Verlag, 2002.

- J. Horwitz and B. Lynn, "Toward Hierarchical Identity-Based Encryption", Proceedings of EU-ROCRYPT 2002, LNCS 2332, Springer-Verlag, pages 466-481, 2002.

- Perry Ismangil and Benny Prijono, *PJSIP: Open Source SIP stack project, 2003*, http://www.pjsip.org/

- ITU-T group, *Packed Based Multimedia Communication Systems, 2006,* http://www.itu.int/rec/T-REC-H.323-200606-I/en

- A. Joux, "One Round Protocol for Tripartite Diffie-Hellman", Algorithmic Number Theory Symposium - Proceedings of ANTS 2002, LNCS 1838, pages 385-394, Springer-Verlag, 2000.

- S. Kent and K. Seo, *Security Architecture for the Internet Protocol, 2005,* http://tools.ietf.org/html/rfc4301

- Lei Kong, Vijay Arvind Balasubramaniyan and Mustaque Ahamad, "A Lightweight Scheme for Securely and Reliably Locating SIP Users", The 1st IEEE Workshop on VoIP Management and Security, 2006.

- John Linn, *Privacy Enhancement for Internet Electronic Mail: Part I : Message Encipherment and Authentication Procedures, 1987*, http://tools.ietf.org/html/rfc989

- Ben Lynn, *Pairing Based Cryptography Library, 2005,* http://crypto.stanford.edu/pbc/

- J. Malone-Lee, *Identity Based Signcryption, 2003*, http://eprint.iacr.org/2002/098/.

- S. McGann and Douglas C. Sicker, "An Analysis of Security Threats and Tools in SIP-Based VoIP Systems", Proceedings of the 2nd Workshop on Securing Voice over IP, Washington, June 2005.

- Alfred J. Menezes, Tatsuaki Okamoto and Sott A. Vanstone, "Reducing Elliptic Curve Logarithms to Logarithms in a Fint Field", IEEE, Transactions on Information Theory, 1993.

- Takeshi Okamoto, Raylin Tso, and Eiji Okamoto, "One-way and two-party authenticated id-based key agreement protocols using pairing", Modeling Decisions for Artificial Intelligence, Second International Conference, MDAI 2005, Tsukuba, Japan, July 25-27, 2005, Proceedings, 3558/2005: 122-133, 2005.

- Qi Qiu, *Study of Digest Authentication for SIP*, Master's Project, University of Ottawa, 2003.

- E. Rescorla, *Diffie-Hellamn Key Agreement Method, 1999,* http://www.ietf.org/rfc/rfc2631.txt

- R. Rivest, *The MD5 Messaeg-Digest Algorithm, 1992,* http://www.ietf.org/rfc/rfc1321.txt

- Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM 21 (2), pages 120-126, 1978.

- Kenneth H. Rosen, *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, Chapman and Hall/CRC, USA, 2006.

- J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "Sip: Session Initiation Protocol," RFC 3261, Internet Engineering Task Force, 2002.

- S. Salsano, L. Veltri, D. Papalilo, "SIP Security Issues:The SIP Authentication Procedure and its Processing Load", IEEE Network, Vol. 16, Iss. 6, p.38- 44, Nov/Dec 2002.

- H. Schulzrinne, S. Carner, R. Fredrick and V. Jacobson, *RTP: Transport Protocol for Real-Time Applications, 2003,* http://www.ietf.org/rfc/rfc3550.txt.

- Jan Seedorf, "SIP Security Status Quo and Future Issues", Security in Distributed Systems (SVS), 2007.

- A. Shamir, “Identity-based Cryptosystems and Signature Schemes”, Proceedings of CRYPTO '84, LNCS 196, pages 47-53, Springer-Verlag, 1984.

- K. Singh and H. Schulzrinne, "Peer-to-peer Internet Telephony using SIP," Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video NOSSDAV'05, Stevenson, Washington, USA, pp. 63-68, 2005.

- N.P. Smart, “An identity based authenticated key agreement protocol based on the Weil pairing”, Electronics Letters, 38:630-632, 2002.

- W. Stallings, Cryptography and Network Security: Principles and Practices, 4th edition, Prentice Hall, 2006.

- Voice System team, *OpenSER project, 2005*, http://www.opensips.org/

- Xiaoyun Wang and Hongbo Yu, “How to Break MD5 and Other Hash Functions”, EUROCRYPT, Shandong University, China, 2005.

- Lawrence C. Washington, *Elliptic Curves Number Theory and Cryptography*, Chapman and Hall/CRC, Maryland, 2008.

- Eun-Jun Yoon ad Kee-Young Yoo, “Simple Authenticated Key Agreement Protocol based on Weil Pairing”, IEEE Computer Society, International Conference on Convergence Information Technology, p.2096 - 2101, 2007.