第 35 卷 第 2 期 计 算 机 工 程 2009 年 1 月
Vol.35 No.2 Computer Engineering January 2009
·安全技术· 文章编号：1000—3428(2009)02—0162—02 文献标识码：A 中图分类号：TP393

# 基于 SIP 的安全认证机制的研究及改进

李 婧，李 雪，胡 浩
(解放军信息工程大学信息工程学院，郑州 450002)

**摘 要：**会话初始协议大部分认证机制只提供服务器到客户端的单向认证，HTTP 摘要认证就是其中的一种。该文通过分析其过程，找出认证协议中的安全缺陷，给出攻击者可能进行的攻击。针对协议的安全漏洞，提出一种改进的安全机制，在提供服务器和客户端之间相互认证的基础上加入加密保护和完整性保护，以保证消息传输的安全性。

**关键词：**会话初始协议；认证；HTTP 摘要；安全

# Research and Improvement of Secure Authentication Scheme Based on Session Initial Protocol

LI Jing, LI Xue, HU Hao
(Institute of Information Engineering, PLA Information Engineering University, Zhengzhou 450002)

**【Abstract】**Most Session Initial Protocol(SIP) authentication schemes only provids client-to-server authentication, and HTTP Digest authentication is one of them. This paper analyzes the procedure and security of HTTP Digest authentication, and describes the vulnerability and possible attacks to the protocol. According to the vulnerability and possible attacks, it presents an improved scheme, which achieves secure transfer of message, and analyzes its security.

**【Key words】**Session Initial Protocol(SIP); authentication; HTTP Digest; security

## 1 概述

会话初始协议(Session Initial Protocol, SIP)是由IETF提出并主持研究的一种基于应用层的多媒体会话控制协议，它是实现新一代多媒体通信和软交换的关键技术，正成为VoIP通信的主流协议之一。面对复杂、开放的网络应用环境，采用文本形式表示消息词法和语法的SIP协议更容易被攻击者模仿、篡改，从而被非法利用。因此，SIP安全变得越来越重要。SIP和IP电话标准化过程中的一个热点问题就是如何加强安全支持以满足业务需要。当客户机请求得到SIP服务时，它需要首先得到服务器的认证。基于以上的原因，RFC2543 提出了HTTP摘要(HTTP Digest)和HTTP Basic 2 种认证方式。由于HTTP Basic方式在消息明文中直接传送密钥，因此密钥很容易在传送过程中被窃取，存在很大的安全隐患，在RFC3261中这种方式已被废除。在HTTP Digest方式中，服务器采用一种基于挑战响应的方式来验证用户的身份。这种方式虽然不在消息中明文传送密钥，但也存在很大的安全隐患。例如，客户端不能验证服务器的身份，如果攻击者伪装成服务器就能获得用户的一些敏感信息，并且容易遭受离线密钥猜测等攻击[1]。

本文分析了 HTTP Digest 认证方式的流程及存在的缺陷，并参考 3G/UMTS 认证和密钥协商机制的原理，结合 HTTP 摘要认证，提出了一种既能实现服务器和客户端之间的双向认证，又能比较有效地避免伪装用户等攻击的高效的安全策略。

## 2 HTTP Digest认证[1]

### 2.1 认证流程

认证开始前，客户端和服务器预先共享一个密钥。服务器用这个密钥验证客户端的身份。认证流程如图 1 所示。

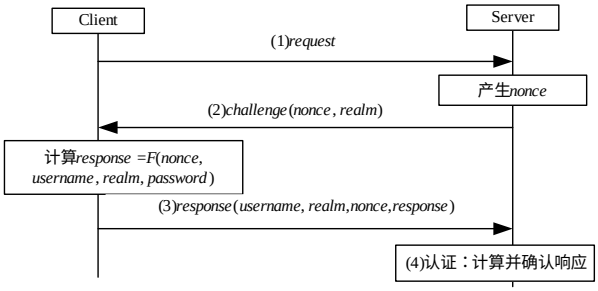


**图 1 HTTP Digest 认证流程**

(1)客户端→服务器：客户端发送一个请求到服务器。

(2)服务器→客户端：服务器要验证客户端身份的合法性，向客户端回送一个包含 *nonce* 和 *realm* 的摘要盘问消息，即 *challenge*。*nonce* 是服务器产生的一个包含时间戳信息的随机数串，*realm* 通常指服务器所负责管辖域的域名。

(3)客户端→服务器：客户端根据收到的 *nonce*, *realm* 及自己的用户名和共享密钥，通过一个单向哈希函数 *F* 计算 *response*：*response*= *F*(*nonce*, *username*, *realm*, *password*)。通常 *F* 用于产生一个摘要认证信息，多数情况下用的是 MD5 算法。然后客户端将 *response* 值发送给服务器。

(4)通过用户名，服务器在数据库中提取出密钥，然后检验 *nonce* 是否正确。如果正确，服务器计算 *F*(*nonce*, *username*, *realm*, *password*)，将结果与收到的 *response* 做比较，如果匹配，服务器就认为客户端是一个合法的用户。

**作者简介：**李 婧(1979－)，女，博士研究生，主研方向：移动通信；李 雪、胡 浩，硕士研究生

**收稿日期：**2008-04-10 **E-mail：**ljatzz_1997@163.com

### 2.2 认证存在的缺陷

(1)离线密钥猜测攻击

在步骤(2)和步骤(3)中，攻击者通过截获消息，可以很容易地获得 *nonce, username, realm* 及 *response* 的值，接着攻击者猜测密钥 *password'*，并计算 *F*(*nonce, username, realm, password'*)，将结果与 *response* 值进行比较，如果匹配，*password'*便是正确的密钥。

(2)服务器伪装攻击

如果用户不验证服务器的身份是否合法，当它收到服务器发来的 *challenge* 消息时，就会回复一个 *response*。这样一个攻击者可以伪装成服务器的身份向用户发送 *challenge* 以获得 *response*，并通过对多次伪装服务器收到的多个 *response* 值进行分析，以离线猜测出用户的正确密钥。

## 3 改进的安全认证

### 3.1 安全认证存在的缺陷

文献[1]提出了一种改进的安全机制，可以实现客户端和服务器的相互认证，弥补了离线密钥猜测攻击和服务器伪装攻击的缺陷。但这种安全机制无法防止伪装用户的攻击。可能的攻击如下：

**步骤 1** 客户端→攻击者：伪装用户的攻击者截获客户端的合法请求。

**步骤 2** 攻击者→服务器：伪装用户的攻击者将截获客户端的合法请求发送给服务器。

**步骤 3** 服务器→攻击者：服务器向伪装用户的攻击者发送正确的挑战消息。

**步骤 4** 攻击者→客户端：伪装用户的攻击者向客户端转发正确的挑战消息。

**步骤 5** 客户端→攻击者：客户端发送正确的响应消息。

**步骤 6** 攻击者→服务器：伪装用户的攻击者向服务器转发正确的响应消息。

这样，伪装用户的攻击者就可以假冒该合法客户端入网。

### 3.2 改进的安全认证流程[2]

改进的安全机制除了保留客户端和服务器之间的相互认证，还加入了加密保护和完整性保护以避免伪装用户的攻击。

对消息完全加密是保证消息机密性最可靠的方式，理论上SIP协议可以通过底层的安全机制保证其安全，如通过网络层的IPSec或传输层的TLS对SIP消息实现完全加密。由于IPSec网络实施复杂、实现代价比较高，而TLS可能遭受IP欺骗[3]，因此本文的安全机制在客户端和服务器相互认证的同时产生加密密钥，并在以后的信令交互中使用该加密密钥对信令及数据进行加密保护。

完整性保护指数据在未授权情况下不能被修改或删除，同时数据的来源是合法的，必须保证信息的完整性。在SIP中，如果接收端和请求端某些头域不完全相等，就是破坏了完整性，必须通知使用者这个改变[4]。

认证开始前，服务器端和客户端预先共享一个密钥 *K* 和一个比较大的认证序号 *SEQ*(*SEQ* 以递增的顺序进行)，认证流程(图 2)如下：

(1)客户端→服务器：客户端发送一个请求到服务器。其中除了用户名还包括客户端支持的加密算法 *UEA* 和完整性算法 *UIA*。

(2)服务器→客户端：服务器生成一个随机数*nonce*，并计算 2 个参数 *AK* 和 *AM* 的值：*AK=F*(*nonce, username, K*)，*AM=F*(*nonce, SEQ, K*)，*AC*为*AK*⊕*SEQ*与*AM*连接后的结果，这种连接方式为服务器和客户端预先设定好一种方式，比如在*AC*中，从密钥长度位开始的一部分为*AM*，其余字段为*AK*⊕*SEQ*。*F*仍表示单向哈希函数。同时，服务器根据随机数*nonce*和共享密钥*K*生成并储存加密密钥*CK*、完整性密钥*IK*以及期望响应*XRES*，$f_{CK}$,$f_{IK}$ 以及$f_{RES}$分别是其对应的密钥生成函数。根据客户端支持的加密算法、完整性算法和服务器支持的加密算法、完整性算法，选择相匹配的算法*UEA'*和*UIA'*。然后服务器向客户端发送挑战消息，其中包括*nonce, username, AC* 3 个参数的值、选择的算法*UEA'*和*UIA'*以及在步骤(1)请求消息中传送的算法*UEA*和*UIA*。注意：这条消息开始用刚生成的完整性密钥*IK*和选择的完整性算法*UIA'*进行完整性保护。

(3)客户端→服务器：客户端根据收到的*nonce*及自己的用户名和共享密钥*K*，通过一个单向哈希函数*F*计算*AK'=F*(*nonce, username, K*)，并与收到的*AK*⊕*SEQ*进行异或计算，得到*SEQ'=AK'*⊕*AK*⊕*SEQ*，验证*SEQ'*的正确性。如果收到的*SEQ'*≠*SEQ*，说明服务器产生的挑战消息不是最新的，认证失败；否则，继续验证服务器身份，计算*AM'=F*(*nonce, SEQ, K*)，比较*AM'*与收到的*AM*是否相等，如果不相等，说明服务器身份不合法，拒绝向服务器发送*response*消息；否则，认为服务器是可信任的，计算*RES*=$f_{RES}$(*nonce, K*)。接着向服务器发送*response*消息，其中包含*username*和*RES*值，并根据随机数*nonce*和共享密钥*K*生成加密密钥*CK*和完整性密钥*IK*。同时应将收到的受完整性保护的*UEA*和*UIA*与自己原先发送的未受完整性保护的算法*UEA*和*UIA*相比较。注意：*response*消息开始用刚生成的完整性密钥*IK*和收到的完整性算法*UIA'*进行完整性保护。

(4)通过用户名，服务器在数据库中提取出存储的 *XRES*，然后与收到的 *RES* 比较，如果匹配，服务器就认为客户端是一个合法的用户。同时对收到的消息进行完整性验证。以后的消息都可以开始加密和完整性保护。

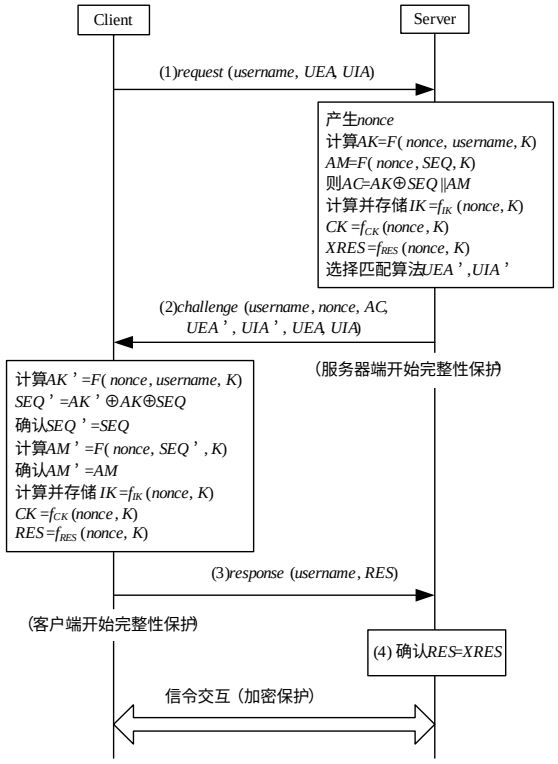


**图 2 改进的安全认证流程**

(下转第 166 页)