A Multifactor Hash Digest Challenge-Response Authentication Scheme for Session Initiation Protocol

S. Santhosh Baboo
Reader in Computer Science, D.G. Vaishnav College
Arumbakkam, Chennai-600 106, Tamilnadu. India.
E-mail: santhos1968@gmail.com

K. Gokulraj
Research Scholar, Centre for Research
Bharathiar University, Coimbatore-641 046, Tamilnadu. India.
E-mail: gokulrajk@yahoo.co.in

Received: November 14, 2010 Accepted: December 25, 2010 DOI: 10.5296/npa.v2i4.512

Abstract

Authentication is a process by which the sender and the receiver identify the legal communicating partners prior to commencement of message transactions. Authentication is a part of security which is ascertained at the time of initiation of the communication between the two communicating entities like client and server. Network communications are found to be vulnerable due to the increase in number of threats from unknown intruders. Session Initiation Protocol (SIP) is used for initializing the session between two communicating devices or entities. This protocol is widely used in multimedia communications. SIP is a powerful signaling protocol which initializes, establishes, maintains, and terminates the session between the communicating devices. Many authentication schemes were proposed for SIP from time to time. Here, we propose a new authentication scheme based on Multifactor Hash Digest Challenge-Response Sequence Count method for SIP. This method enhances the SIP authentication and overcomes vulnerability attacks like Password guessing, Server spoofing, Replay, Bucket Brigade and Modification Attacks. This method enhances the Authentication, Efficiency, Integrity, Reliability, and Security in SIP Authentication process.

Keywords: Challenge-Response, Authentication, Session Initiation Protocol, Multifactor Sequence Count Mechanism, Network Security.

1. Introduction

The SIP is a Session Initiation Protocol standard for Internet Protocol Telephone introduced by the IETF (Internet Engineering Task Force) [1]. Initially this protocol was used in VOIP (Voice Over Internet Protocol) and later it was started to use in multimedia communication over the internet [2]. The SIP is an application layer control signaling protocol which can create, initialize, modify, maintain, and terminate a session between the two or more communicating devices of the corresponding parties [3]. The SIP session performs multimedia conferences, Internet telephone calls, and distribution of multimedia contents [4]. The basic authentication scheme provided by the SIP is obtained from Hyper Text Transfer Protocol (HTTP) digest authentication [5]. The SIP authentication scheme uses, Challenge-Response mechanism in order to identify the legal user [6]. The basic SIP authentication scheme is vulnerable due to many of the emerging threats and attacks like Password guessing, Sever spoofing, Bucket Brigade, Modification, and Replay attacks. Several SIP authentication schemes have been proposed by the researchers from time to time to overcome some of the vulnerable attacks. [7] proposed an authentication scheme in 2005 based on Diffie-Hellman key exchange [8] algorithm depends on the DLP (Discrete Logarithm Problem) to overcome the threats like off-line password guessing, and server spoofing attacks. Later, [9] proposed an another efficient authentication scheme in 2005 for SIP based on Elliptic Curve Diffie-Hellman (ECDH) key exchange algorithm to reduce the execution time and to increase the efficiency with the help of elliptic curve public key pair exchange mechanism which overcomes password guessing, server spoofing attacks. Then, [10] proposed a nonce based authentication scheme in 2009 which uses one-way hash function and XOR operation to reduce the computational cost and to increase the computational performance. Then, [11] proposed a dynamic authentication scheme for the smart card based networks in 2010 which uses Hash Function, XOR operations, and Encryption for enhancing the security features like authentication, confidentiality, reliability, integrity and security during the dynamic authentication process. The proposed scheme overcomes password guessing, server spoofing, replay, bucket brigade, and modification attacks which enhances the security features in authentication process.

This paper proposes an improved challenge-response authentication scheme for SIP. This authentication scheme introduces a method namely Multifactor Hash Digest Challenge-Response Sequence count. The multifactor authentication mechanism includes the factors like UAC Password (PWC), UAC Password Index (IPWC), UAS Password (PWS), UAS Password Index (IPWS), UAC Date of Registration (UDR), UAC Date of Registration Index (IDR), Sequence Count (CS), UAC Session Key (SKC), and UAS Session Key (SKS). The sequence count mechanism is a technique in which the Sequence Counter starts its count from '1' at the very first time between the agreed upon communicating devices. The Sequence Count value is incremented by a pre-determined shared value between committed devices for each step of transaction and the sequence count is updated at sequence count history table. The count value continues in future communication also and the sequence counter is reset only when it reaches it maximum limit. The sequence count mechanism enhances the authentication between the committed communicating parties. In addition to that, this method uses session keys SKC and SKS for the client and server respectively. The session key is valid only for the current session and these keys are generated at the respective side based on pre-determined shared formula. This proposed method prevents some of the active and passive attacks. This method uses Hash Digest and XOR operations. In Multifactor mechanism, appropriate factors are XORed together and they are transformed into One-way hash function (hf). The transformed hash function components are called Hash Digest which is an irreversible message digest.

2. Proposed Authentication Scheme for SIP

The proposed authentication scheme is shown in the Fig.1. This scheme comprises the Setup Phase and Authentication phase. Table 1 shows the notations and their corresponding meaning used in this scheme.

2.1 Setup Phase

This is a phase in which the UAC and the UAS agree with some of the parameters and formula for session key generation before the commencement of authentication phase. Here, UAC chooses its own password namely PWC, and the UAS chooses its own password namely PWS and both of these passwords are mutually shared along with their index values namely IPWC and IPWS respectively to each other by means of any secured mode other than network communication. Similarly, UAC's Identity (UID) and Date of Registration (UDR) are intimated to the UAS. In turn, the UAS intimates SID, IPWC, IDR values to the UAC. These values are separately maintained both by the UAC and UAS for further steps of authentication process. Whenever, the UAC wants to communicate with the UAS, it has to respond with the above mentioned factors along with their index values for the proper authentication steps.

Table 1. Notations and Meaning

Notations	**Meaning**
UAC	User Agent Client
UAS	User Agent Server
SKC	Client Session Key
SKS	Server Session Key
CS	Sequence Count
UID	User Identity Code
SID	Server Identity Code
PWC	Client Password
PWS	Server Password
IPWC	Client Password Index
IPWS	Server Password Index
UDR	User Date of Registration
IDR	Date Registration Index
$\oplus$	Exclusive-OR operation
-	Not Mentioned
hf( )	Hash Function

2.2 Authentication Phase

In this phase, when the UAC wishes to login with the UAS, it enters the identity (UID), Password (PWC) along with its Index (IPWC), Client Session Key (SKC) and Sequence count (CS). The complete steps of authentication are shown in Fig.1. These steps of authentication are given as follows:

Step 1:

UAC $\xrightarrow{\textbf{Request}(U_{ID},\ hf[(I_{PWC} \oplus PW_C) \oplus (SK_C \oplus C_S)])}$ UAS

First, the UAC computes the (IPWC $\oplus$ PWC) component and (SKC $\oplus$ CS) components separately, and both of these components are XORed together to obtain a unified component. Then, these components are transformed into one-way hash digest, and it sends this hash digest along with identity as Request scheme to the UAS. This step performs 3 XOR operations and 1 hash computation.

Step 2:

$$\text{UAS} \xrightarrow{\textbf{Challenge}(\text{Realm},\ hf[(I_{PWS} \oplus PW_S) \oplus (SK_S \oplus C_S)])} \text{UAC}$$

The UAS receives the Request and computes hf[(IPWC$\oplus$PWC)$\oplus$(SKC$\oplus$SC)] digest from the pre-determined shared values, session key generation formula and the history of sequence count table. The SKC is generated by the client for each session of authentication with the server. This key is computed by the server based on the pre-determined shared formula with the client. Using this key the server could reconstruct the received Request digest for comparison. Then, it compares the computed digest with the received digest, and if the computed value does not agree with the received Request, then it discards the Request and further steps of authentication are stopped. If both the values agree with each other, then, UAS realizes that the Request comes from legal UAC. Then, computes Challenge(Realm, hf[(IPWS$\oplus$PWs)$\oplus$(SKS$\oplus$CS)]) and sends it to the UAC. This step performs 3 XOR operations and 1 hash computation.

Step 3:

$$\text{UAC} \xrightarrow{\textbf{Response}(U_{ID},\ \text{Realm},\ hf[(I_{DR} \oplus DR) \oplus (SK_C \oplus C_S)])} \text{UAS}$$

The UAC receives the Challenge, and computes hf[(IPWS$\oplus$PWs)$\oplus$(SKS $\oplus$CS)] digest from the pre-determined shared values, session key generation formula, and sequence count table. The SKS is generated by the server for each session of authentication with the client. This key is computed by the client based on the pre-determined shared formula with the server. Using this key, the client could reconstruct the received Challenge for comparison. The computed Challenge is compared with the received hash digest Challenge. If both the values do not agree, then the UAC discards the Challenge and further steps of authentication between them are stopped. If both the values agree, then the UAC computes Response(UID, Realm, hf[(IDR$\oplus$DR)$\oplus$(SKC$\oplus$CS)]) and sends it the UAS. This step performs 3 XOR operations and 1 hash computation.

Step 4:

$$\text{UAS} \xrightarrow{\textbf{Authentication}(\text{Verifier},\ hf[(I_{PWC} \oplus PW_C) \oplus (I_{PWS} \oplus PW_S) \oplus (I_{DR} \oplus DR) \oplus (SK_S \oplus C_S)])} \text{UAC}$$

The UAS receives the Response and it computes hf[(IDR$\oplus$DR)$\oplus$(SKC$\oplus$CS)] digest from the pre-determined shared values, session key generation formula, and sequence count table. If the computed value does not agree with the received Response, then the UAS rejects the Response. If both values agree each other, then it computes Authentication(Verifier, hf[(IPWC$\oplus$PWC)$\oplus$(IPWS$\oplus$PWS)$\oplus$(IDR$\oplus$DR)$\oplus$(SKS$\oplus$CS)]) and sends it to the UAC.

This step performs 7 XOR operations and 1 hash computation. Thus, this scheme totally performs 16 XOR operations and 4 hash computations.

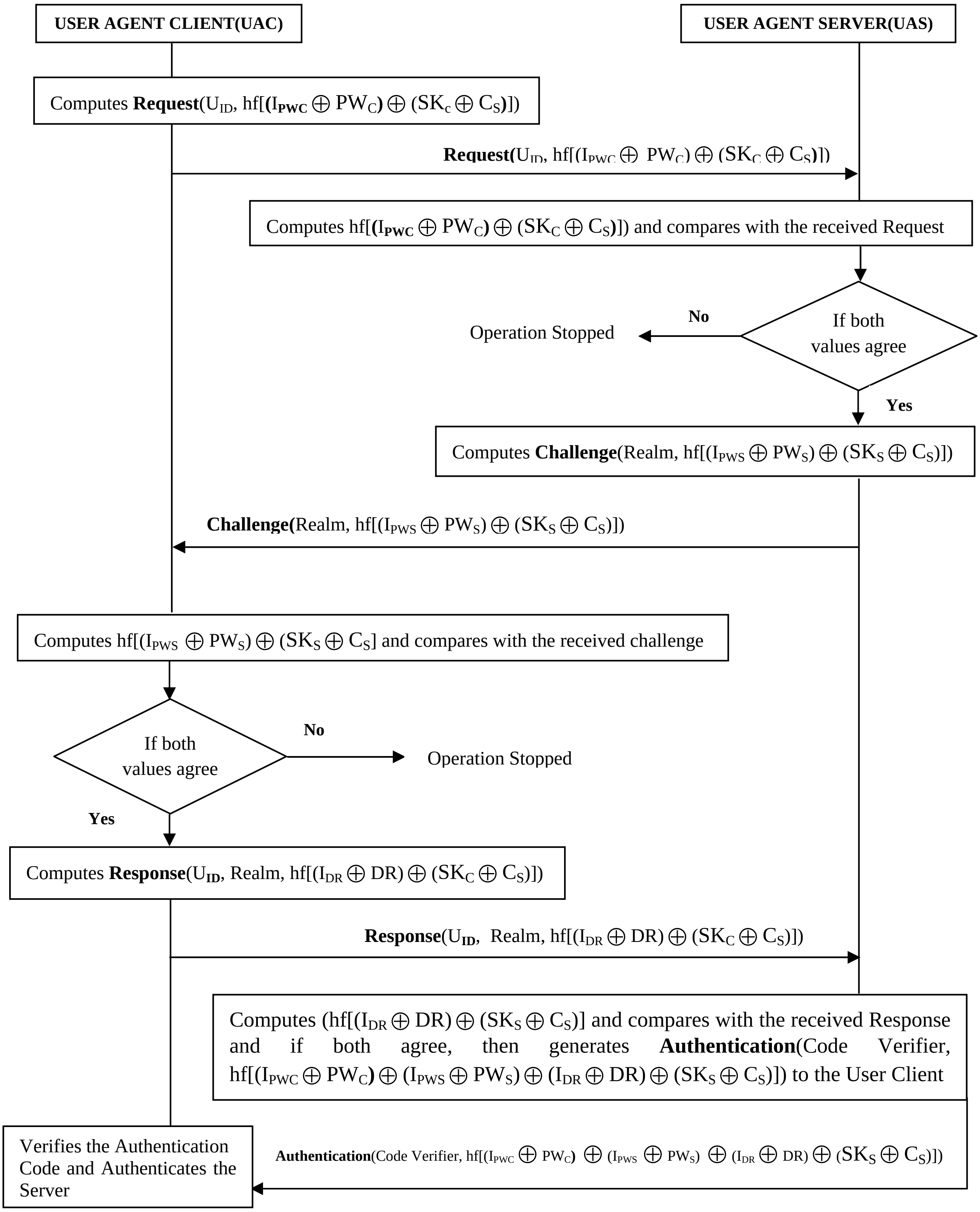
USER AGENT CLIENT(UAC)
USER AGENT SERVER(UAS)
Computes **Request**(U_{ID}, hf[($I_{PWC} \oplus PW_C$) $\oplus$ ($SK_c \oplus C_S$)])
Request(U_{ID}, hf[($I_{PWC} \oplus PW_C$) $\oplus$ ($SK_C \oplus C_S$)])
Computes hf[($I_{PWC} \oplus PW_C$) $\oplus$ ($SK_C \oplus C_S$)]) and compares with the received Request
If both values agree
No
Operation Stopped
Yes
Computes **Challenge**(Realm, hf[($I_{PWS} \oplus PW_S$) $\oplus$ ($SK_S \oplus C_S$)])
Challenge(Realm, hf[($I_{PWS} \oplus PW_S$) $\oplus$ ($SK_S \oplus C_S$)])
Computes hf[($I_{PWS} \oplus PW_S$) $\oplus$ ($SK_S \oplus C_S$] and compares with the received challenge
If both values agree
No
Operation Stopped
Yes
Computes **Response**(U_{ID}, Realm, hf[($I_{DR} \oplus DR$) $\oplus$ ($SK_C \oplus C_S$)])
Response(U_{ID}, Realm, hf[($I_{DR} \oplus DR$) $\oplus$ ($SK_C \oplus C_S$)])
Computes (hf[($I_{DR} \oplus DR$) $\oplus$ ($SK_S \oplus C_S$)] and compares with the received Response and if both agree, then generates **Authentication**(Code Verifier, hf[($I_{PWC} \oplus PW_C$) $\oplus$ ($I_{PWS} \oplus PW_S$) $\oplus$ ($I_{DR} \oplus DR$) $\oplus$ ($SK_S \oplus C_S$)]) to the User Client
Authentication(Code Verifier, hf[($I_{PWC} \oplus PW_C$) $\oplus$ ($I_{PWS} \oplus PW_S$) $\oplus$ ($I_{DR} \oplus DR$) $\oplus$ ($SK_S \oplus C_S$)])
Verifies the Authentication Code and Authenticates the Server

Figure 1. Multifactor Hash Digest Sequential Count Authentication

3.0 Security Analysis

3.1 Off-line password guessing attack

According to the proposed scheme, an attacker cannot guess the off-line password. Since, in this scheme, we introduce two passwords namely UAC password(PWC) and UAS password(PWS) along with their corresponding password index values IPWC and IPWS, which are separately maintained both by the UAC and UAS respectively. When an attacker tries to guess the password, the attacker has to guess these two passwords along with their index values. Though the attacker guesses the UAC's password, it is difficult to guess the index value(IPWC). Similarly, if the attacker tries to guess the UAS's password, it is difficult to guess the index value (IPWS). Only when both the passwords and index values are correctly guessed, the attacker could complete the process of password guessing. But, it is not possible to guess all these factors by the attackers. Because, if the attacker tries to know UAC's password PWC, from the Request(UID, hf[(IPWC $\oplus$ PWC) $\oplus$ (SKC $\oplus$ CS)]) step, the attacker must also know the index value IPWC which is XORed with the PWC. In addition to that, the attacker has to know the client session key(SKC) and the sequence count value SC. To determine either SKC or SKS, the other component should have been known, which is already XORed with one another. The sequence count CS varies in its value for each step of transaction based on the pre-determined count step, and determination of the sequence count by the attacker is also a difficult one. Since the (SKC $\oplus$ CS) component is XORed with the (IPWC $\oplus$ PWC) component and both of them are transformed into one-way hash digest, it is not possible for the attacker to predict and compute Password, Index value, Session Key and Sequence count values. Hence, this method overcomes the off-line password guessing attack.

3.2 Server Spoofing

Similarly, the attacker cannot act as a UAS to deceive the UAC. Because, if the attacker tries to spoof the UAS, the attacker has to respond to the UAC with the Challenge (Realm, hf[(IPWS $\oplus$ PWS) $\oplus$ (SKS $\oplus$ CS)]). For computing challenge, the attacker must know all the pre-determined shared values, session key generation formula, and sequence count. Since each factor of the above is XORed with one another and transformed into hash digest, determination of SKS or SC is also a difficult process by the attacker. So, unless the attacker knows the current sequence count, and session key, it is very difficult to perform server spoofing. However, if the attacker observed the past transactions between the UAC and UAS, and if tries to make use of the recorded Challenge(Realm, hf[(IPWS $\oplus$ PWS) $\oplus$ (SKS $\oplus$ CS)]) scheme and sending it to the UAC, this scheme is not accepted by the UAC. Because, the UAC computes the received Challenge from its pre-determined shared values, session key generation formula, and sequence count history table. Then it compares both the values and if these values do not agree, the UAC rejects the Challenge. Thus, this scheme prevents the server spoofing attack.

3.3 Replay Attack

In this type of attack, an attacker or Evesdropper may intercept and Replay any one of the steps of authentication like Request, Challenge, Response, or Authentication between the UAC and UAS from the past observations. If the Eve tries to Replay the Request(UID, hf[(IPWC $\oplus$ PWC) $\oplus$ (SKC $\oplus$ CS)]) step to the UAS, it is detected by the UAS by comparing the received Request with the computed Request. Both the values do not agree upon comparison. Since, the session key and sequence count value vary for each step of transaction; the UAS rejects the Request from the Eve. If the Eve tries to replay the Challenge(Realm, hf[(IPWS⊕PWS)⊕(SKS⊕CS)]) step to the UAC, the UAC compares the received Challenge with the computed Challenge and it determines that both the values do not agree due to the improper session key and sequence count. Similarly, if the Eve Replays the Response(UID, Realm, hf[(IDR⊕DR)⊕(SKC⊕CS)]) step or Authentication(Verifier, hf[(IPWC⊕PWC)⊕(IPWS⊕PWS)⊕(IDR⊕DR)⊕(SKS⊕CS)]) step, the UAS and the UAC verify these schemes and determine that there are variations in session key and sequence count value. So, the UAS and UAC reject the received steps of authentication. Thus, this method prevents the Replay attack. Since this method prevents the replay attack, the reliability of the authentication scheme is enhanced.

3.4 Bucket Brigade attack

This attack is also called as Man-in-the-middle attack. In the proposed scheme, this attack is not possible for an attacker by acting either as UAS or UAC to deceive the other side of authentication process. If the attacker deceives either UAC or UAS, the attacker has to know mutual password of UAC and UAS. Since this scheme introduces independent passwords mutually shared between the UAC and UAS along with their index values, the attacker cannot determine the password, index value, session key, and sequence count value from the transformed one-way hash digest. If the attacker intercepts the Request(UID, hf[(IPWC⊕PWC)⊕(SKC⊕CS)]) step, the UAS computes the Request from the mutually shared values, session key generation formula, and sequence count table and then compares the computed Request with the received Request. If both the values do not agree, then the UAS rejects the Request. Similarly, if the attacker intercepts the Challenge(Realm, hf[(IPWS⊕PWS)⊕SKS⊕CS)]) step, the UAC computes the Challenge step from the mutually shared values, key generation formula, and sequence count history table. Then, it compares both the values and if these values do not agree, then the UAC rejects the Challenge. Thus, this method resists the Bucket Brigade attack or Man-in-the-middle attack. The prevention of interceptions in this method ensures the integrity in authentication process.

3.5 Modification Attack

This is a type of attack using which an attacker may try to modify the contents of the any steps of authentication process. But, this method prevents this type of attack by means of reverse computation and comparison technique of the received digest values from the known values of respective factors. Since this scheme verifies the received component from another side at each step of authentication, any modification made in any step of authentication is easily identified and then further steps of authentication are immediately stopped. If the Eve modifies any one of the steps like Request(UID, hf[(IPWC⊕PWC)⊕(SKC⊕CS)]) or Challenge(Realm, hf[(IPWS⊕PWS)⊕(SKS⊕CS)]) or Response(UID, Realm, hf[(IDR⊕DR)⊕(SKC⊕CS)]) or Authentication(Verifier, hf[(IPWC⊕PWC)⊕(IPWS⊕PWS)⊕(IDR⊕DR)⊕(SKS⊕CS)]), the receiving end verifies this value with the computed values of the parameters. If any discrepancy found upon comparison, then the step is rejected and no further steps of authentication is permitted. Thus, this method prevents the modification attack. Since, this scheme prevents the modification attack, it enhances the integrity and reliability in authentication process.

4. Discussions

Table 2 shows the different types of attacks dealt by the mentioned authentication schemes. This table infers that the HTTP digest scheme overcomes the Replay attack. But this scheme is vulnerable to the off-line Password guessing attack and Server Spoofing attack. The remaining mentioned attack types were not dealt by this scheme. The Durlanik et al. scheme and Tsai et al. scheme overcome Password guessing attack and Server spoofing attack. But, the remaining mentioned attacks were not dealt by these schemes. The EKE scheme and Yang et al. scheme overcome Password guessing attack, Server spoofing attack, and Replay attack. But, the remaining mentioned attacks were not dealt by these schemes. Our proposed scheme overcomes all the mentioned attack types through the enhanced security in authentication due to multifactor and sequence count mechanisms.

Table 2. Comparison Table for possible attacks dealt by different schemes

Attack Types	**HTTP Digest Scheme**	**Durlanik et al. Scheme**	**Tsai Scheme**	**EKE Scheme**	**Yang et al. Scheme**	**The Proposed Scheme**
Password guessing attack	Yes	No	No	No	No	No
Server Spoofing Attack	Yes	No	No	No	No	No
Replay Attack	No	-	-	No	No	No
Bucket Brigade Attack	-	-	-	-	-	No
Modification Attack	-	-	-	-	-	No

Table 3 shows the Method, Operations, and Security mechanisms dealt by the mentioned authentication schemes. This table shows that the HTTP digest scheme uses only one hash computation. The EKE scheme uses 4 exponentiation and 9 symmetric encryption operations. Yang et al. scheme uses 7 hash computation, 4 Exponentiation, and 4 XOR operations. Durlanik et al. scheme uses 7 hash computations, 6 ECC computations, and 2 XOR operations. Tsai et al. scheme uses 7 hash computations, and 4 XOR operations. But, our proposed scheme uses 4 hash computations and 16 XOR operations for enhancing the integrity, reliability, and security in authentication. The enhanced features of this scheme cannot be compromised by any of the mentioned vulnerable attacks.

Table 3. Comparison Table for Method, Operations, and Security dealt by different schemes

Method, Operations, and Security	**HTTP Digest Scheme**	**EKE Scheme**	**Yang et al. Scheme**	**Durlanik et al. Scheme**	**Tsai Scheme**	**The Proposed Scheme**
Method	MD	Encryption	DH	ECDH	HF	Hash Counter
Operations	HF	EXP, ENC	HF, DL, EXP, XOR	HF, DLP, ECC, XOR	HF, XOR, Concatenation	HF, XOR
No. of Hash Computations	1	-	7	7	7	4
No. of ECC Computations	-	-	-	6	-	-
No. of XOR Computations	-	-	4	2	4	16
No. of Exponentiation	-	4	4	-	-	-
Symmetric Encryption	-	9	-	-	-	-
Authentication						Enhanced

Efficiency						Enhanced
Integrity						Enhanced
Reliability						Enhanced
Security						Enhanced

The integrity and reliability enhance the security in authentication process. Since all the steps of authentication are protected effectively by this method from the vulnerable attacks, this method enhances authentication, efficiency, integrity, reliability, and security in authentication for Session Initiation Protocol.

The above table of information is also shown as an analytical graph in Fig.2, which depicts the computations of different schemes and enhanced features of the proposed scheme. In the proposed scheme, the Hash computation and XOR computations only are used, and the increase in XOR computations are meant for emphasizing the integrity of the authentication process. The Integrity enhances the reliability in authentication process.

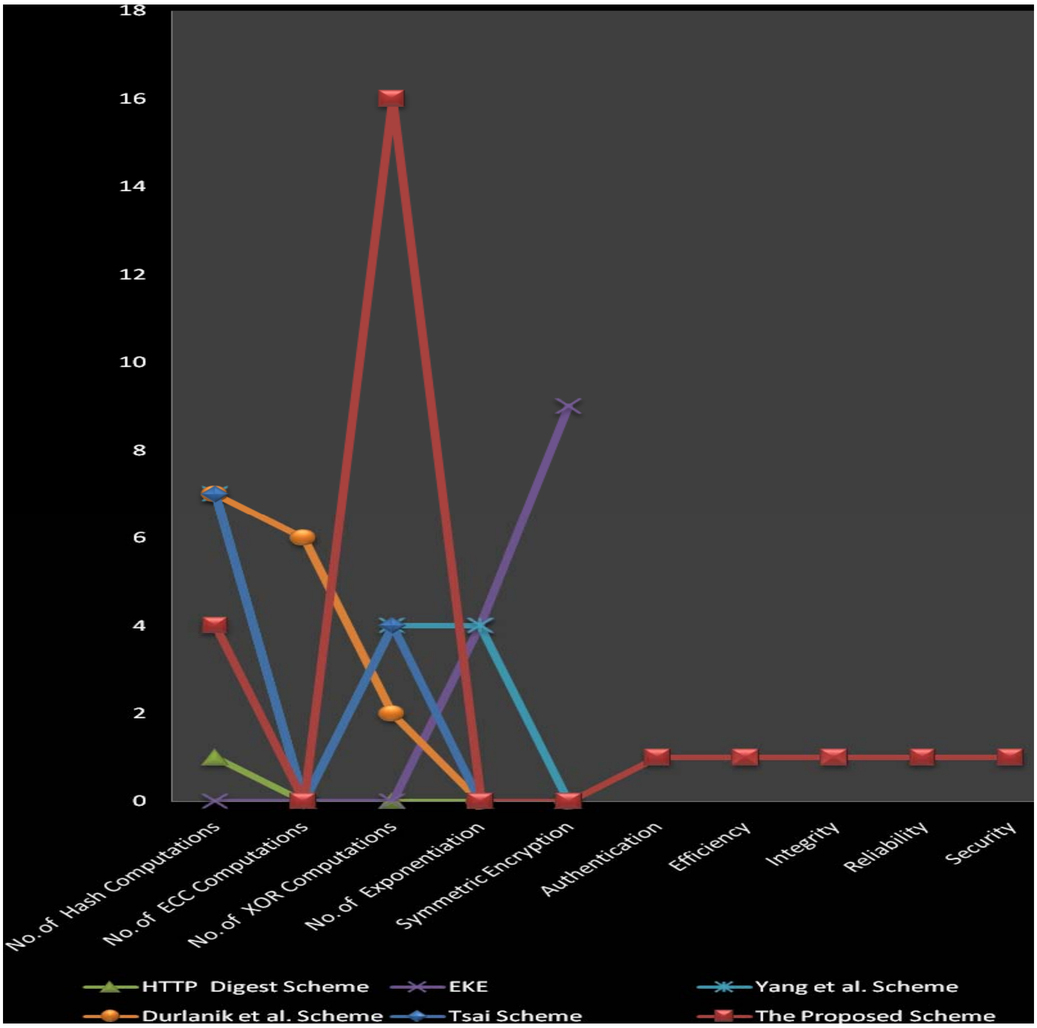

Figure 2. Analytical Graph of Authentication Schemes

5. Conclusion

In this work, we have analyzed the different SIP authentication schemes. The attack types dealt by these authentication schemes have been analyzed. The methods, operations, and security features provided by these authentication schemes have also been analyzed. Our proposed authentication scheme has introduced multifactor hash digest sequence count challenge-response mechanism to enhance the authentication, efficiency, integrity and reliability for SIP. This authentication scheme prevents the Off-line Password guessing attack, Server spoofing attack, Replay attack, Bucket Brigade attack, and Modification attacks. The technique of this scheme emphasizes the security features enhancement in authentication process. Thus the proposed scheme enhances the network security in authentication process for Session Initiation Protocol.

References

[1] Arkko J, et al. Security mechanism agreement for SIP sessions. IETF Internet draft, June 2002.

[2] Franks J., Hallam-Baker P., Hostetler J., Lawrence S. HTTP authentication: Basic and digest access authentication, 2617, IETF Network Working Group, June 1999.

[3] Rosenberg J., Schulzrinne H., Camarillo G., Johnston A., Peterson J., Sparks R., Handley M. and Schooler E. SIP: Session Initiation Protocol. RFC 3261, IETF. The Network Working Group, June 2002.

[4] Lin, C. L. and T. Hwang. “A password authentication scheme with secure password updating,” Computers and Security, vol. 22, no. 1, pp. 68-72, 2003.

[5] Peterson J. Session Initiation Protocol (SIP) Authenticated Identity Body (AIB) Format. RFC 3893, IETF Network Working Group, September 2004.

[6] Veltri L, Salsano S, Papalilo D. SIP security issues: the SIP authentication procedure and its processing load. IEEE Network; pp.16 (6):38e44, 2002.

[7] Yang. C. C., Wang. R. C., Liu. W. T. “Secure authentication scheme for session initiation protocol,” Computers & Security, vol. 24, pp. 381-386, 2005.

[8] Diffie Whitfield, Hellman M. New Directions in Cryptography, IEEE Transactions on Information, Theory: IT:pp.22 (6):644e54:1976.

[9] Durlanik A, and Sogukpinar I. “SIP authentication scheme using ECDH,” World Enformatika Society Transaction on Engineering Computing and Technology, vol. 8, pp. 350-3, 2005.

[10] Tsai. J. L. “Efficient nonce-based authentication scheme for session initiation protocol,” International Journal of Network Security, vol. 8, no. 3, pp. 312-6, May. 2009.

[11] Santhosh Baboo. S, and Gokulraj K. “A Secure Dynamic Authentication Scheme for Smart Card based Networks,” International Journal of Computer Applications, vol. 11, no.8, pp. 5-12, 2010.

Copyright Disclaimer