sensors

MDPI

Article

Securing Session Initiation Protocol

Osama Younes [1,2,*] and Umar Albalawi [1,3]

[1] Faculty of Computer and Information Technology, University of Tabuk, Tabuk 47512, Saudi Arabia
[2] Faculty of Computers and Information, Menoufia University, Shibin El Kom 32511, Egypt
[3] School of Computing & Data Science, Wentworth Institute of Technology, Boston, MA 02115, USA
* Correspondence: o_younes@ut.edu.sa or usama_younas@ci.menofia.edu.eg

Abstract: The session initiation protocol (SIP) is widely used for multimedia communication as a signaling protocol for managing, establishing, maintaining, and terminating multimedia sessions among participants. However, SIP is exposed to a variety of security threats. To overcome the security flaws of SIP, it needs to support a number of security services: authentication, confidentiality, and integrity. Few solutions have been introduced in the literature to secure SIP, which can support these security services. Most of them are based on internet security standards and have many drawbacks. This work introduces a new protocol for securing SIP called secure-SIP (S-SIP). S-SIP consists of two protocols: the SIP authentication (A-SIP) protocol and the key management and protection (KP-SIP) protocol. A-SIP is a novel mutual authentication protocol. KP-SIP is used to secure SIP signaling messages and exchange session keys among entities. It provides different security services for SIP: integrity, confidentiality, and key management. A-SIP is based on the secure remote password (SRP) protocol, which is one of standard password-based authentication protocols supported by the transport layer security (TLS) standard. However, A-SIP is more secure and efficient than SRP because it covers its security flaws and weaknesses, which are illustrated and proven in this work. Through comprehensive informal and formal security analyses, we demonstrate that S-SIP is secure and can address SIP vulnerabilities. In addition, the proposed protocols were compared with many related protocols in terms of security and performance. It was found that the proposed protocols are more secure and have better performance.

Keywords: SIP security; authentication protocols; key agreement; SRP analysis; VoIP

Citation: Younes, O.; Albalawi, U. Securing Session Initiation Protocol. *Sensors* **2022**, *22*, 9103. https://doi.org/10.3390/s22239103

Academic Editor: Valderi R. Q. Leithardt

Received: 27 October 2022
Accepted: 17 November 2022
Published: 23 November 2022

1. Introduction

Voice over internet protocol (VoIP) reduces telecommunication costs and provides benefits to business that legacy telephone systems cannot. VoIP systems need efficient, flexible and secure transmission and signaling protocols. The real-time transport protocol (RTP) [1] is used for transmitting media streams such as voice and video over IP networks. To initiate, preserve, and stop multimedia sessions among participants, VoIP services use the session initiation protocol (SIP), which is a client/server text-based signaling protocol [2]. SIP has been used in different applications, such as video conferences, voice/video distribution, and online games [3].

Due to the rise in VoIP applications, the issue of security in SIP has become pivotal [4,5]. According to the security analysis of [6,7], the major source of attacks on VoIP technology is due to vulnerabilities of SIP. Although SIP is widely used as a signaling protocol, it is vulnerable to several attacks, such as registration hijacking, impersonation, message tampering, eavesdropping, and man-in-the-middle attacks. Such attacks degrade the trust level of users to completely rely on VoIP technologies.

To avoid most SIP threats, the SIP protocol needs to support a number of security mechanisms: authentication, confidentiality, integrity, and availability services [8,9]. Authentication mechanisms are used to prevent different masquerading and spoofing attacks. Attacks related to eavesdropping on signaling, media, or network management sessions

can be avoided using confidentiality mechanisms. The lack of integrity mechanisms allows malicious users to manipulate messages to perform different attacks such as replaying, malformed message, and spoofing attacks [10]. Availability mechanisms protect the availability of entities in VoIP systems. They can prevent different denial-of-service (DoS) attacks and flooding attacks.

SIP security basically relies on well-known internet security standards that support authentication, confidentiality, integrity, and/or availability [2], which are HTTP digest [11], IP security (IPSec) [12], transport layer security (TLS) [13], and secure multipurpose internet mail extensions (S/MIME) [14]. These standard security mechanisms are applied in existing and widely used internet applications and services. However, none of these SIP security solutions are a silver bullet; each of them has many demerits, which are discussed in Section 2.

Numerous authentication and key agreement schemes have been proposed in the literature for SIP. However, most of them did not support confidentiality, integrity, and/or availability. These schemes can be divided into two groups: public key infrastructure (PKI) mechanisms and PKI-free mechanisms. PKI mechanisms [15] can effectively provide mutual authentication and key agreements over public channels. However, these approaches require users to store additional data (e.g., certificates) on their devices, which are vulnerable to physical attacks. Additionally, cryptographic operations during authentication and key exchange processes are computationally expensive. PKI-free mechanisms, which are also known as password-authenticated key exchange (PAKE) mechanisms, are considered a promising alternative solution to these concerns [16]. PAKE mechanisms do not require clients to store any secret information but rather allow them to authenticate with a server by entering the user's password. In addition, they are computationally lightweight cryptographic protocols that make them suitable for VoIP systems.

PAKE schemes are divided into two categories: balanced PAKE (B-PAKE) and augmented PAKE (A-PAKE). In balanced PAKE schemes, to authenticate a user, the server stores a value derived from the user's password, which is used to establish a common secret key between the client and the server. However, these schemes cannot protect the password if the server is compromised, because an attacker can use this value to impersonate the user or the server.

A-PAKE authentication protocols can provide more security than B-PAKE protocols. Nevertheless, they are also more complex and computationally expensive to implement. Many A-PAKE-based authentication protocols designed for general client–server applications have been proposed in the literature: SRP [17], AugPAKE [18], OPAQUE [19], Augmented-EKE [20], B-SPEKE [21], J-PAKE [22], and PAKE2+ [23]. The secure remote password (SRP) protocol was introduced in [24] and is described as SRP-3 in RFC 2945 [25]. SRP-6 is a variant of SRP-3 that is more flexible and secure. As described in RFC 5054 [26], it is used for password authentication in SSL/TLS. Additionally, it is included in IEEE P1363 [27] and ISO/IEC 11770-4 standards.

According to the Stanford SRP homepage [28], "SRP is most widely standardized protocol of its type, and as a result is being used by organizations both large and small, commercial and open-source, to secure nearly every type of human-authenticated network traffic on a variety of computing platforms". The SRP protocol is currently at revision 6a [17,28]. Compared to other A-PAKE protocols, SRP is the more practical choice because it has more available implementations, is built in integration with TLS [26], and is faster and secure enough for common use [17]. Unfortunately, as we prove in Section 4, SRP-6a is vulnerable to offline password guessing attacks, stolen verifier attacks, and Denning–Sacco attacks. Many A-PAKE-based authentication schemes for SIP have been introduced in the literature [29–52]. However, most of these schemes have security vulnerabilities, as explained in Section 2.

To secure SIP, this work introduces a new protocol called secure-SIP (S-SIP). As explained in detail in Section 6, S-SIP thwarts most SIP attacks such as impersonation, replay, man-in-the-middle, session teardown, registration hijacking, request spoofing, message

tampering, and re-invite attacks. S-SIP consists of two protocols: the SIP authentication (A-SIP) protocol and the key management and protection (KP-SIP) protocol. A-SIP is an authentication protocol based on SRP. A-SIP is more secure and efficient than SRP. It not only covers all the security flaws and weaknesses of SRP but also provides more functionalities. To stop impersonation and spoofing attacks, A-SIP is used to secure the registration of the authenticated users to the proxy server and distribute the session keys. KP-SIP is used to exchange session keys between clients and secure the signaling messages of SIP. It provides confidentiality and integrity services for SIP. To evaluate the security attributes of the proposed protocols, the ProfVerif [53] tool is used, which is one of the best tools for automated security analysis of cryptographic protocols. The main contributions of this work are as follows:

1. For the first time, we provide an informal security analysis for the SRP protocol.
2. Based on the SRP protocol, we introduce a new authentication scheme that overcomes vulnerabilities and weaknesses discovered in SRP.
3. A new protocol is introduced to provide confidentiality and integrity services for SIP.
4. The security of the proposed protocols is formally and informally analyzed.
5. The proposed protocols are compared with many related protocols in terms of security and performance.

The rest of the paper is organized as follows: The related work is discussed in Section 2. In Section 3, basic concepts about SIP and SRP protocols are explained. Section 4 presents informal security analysis of the SRP protocol. Section 5 explains the proposed protocol S-SIP. The informal and formal security analyses of S-SIP are discussed in Sections 6 and 7, respectively. Performance analysis and comparison are presented in Section 8. Finally, some conclusions are drawn in Section 9.

2. Related Work

Many security mechanisms have been applied to protect SIP from various attacks. These mechanisms can be divided into two categories: standard-based and research-based solutions. These solutions are discussed in this section.

2.1. Standard-Based Solutions

To secure SIP, well-known internet security mechanisms were adopted. More specifically, the following internet security standards were usually used for securing SIP: HTTP digest, TLS, IPSec and S/MIME.

(1) HTTP Digest

HTTP digest [11] is a simple challenge–response protocol that uses a shared secret along with a username, a domain name, a nonce, and specific fields from the SIP message to compute a cryptographic hash, which is used for entity authentication. An SIP server or a client can challenge the request initiator to send a proof of its identity.

Although the HTTP digest protocol can offer one-way message authentication and replay protection, it does not provide mutual authentication, message integrity or confidentiality [8]. Therefore, it cannot thwart man-in-the-middle (MITM) and impersonation attacks. Moreover, if short or weak passwords are used as a shared secret, the HTTP digest protocol becomes vulnerable to offline dictionary attacks [9].

(2) Transport Layer Security

The transport layer security protocol [13] is defined in RFC 4346. TLS provides security services at the transport layer of the internet architecture. TLS offers authentication, integrity, and confidentiality services among SIP entities. It is widely used on the internet today to secure web browsing. TLS is composed of two protocols: the TLS record protocol and the TLS handshake protocol. The TLS handshake protocol is used for mutual authentication and to negotiate cryptographic properties of the respective session. The TLS record protocol aims to maintain a secure connection between two end points. The TLS

handshake has to be completed successfully before transmitting any data. SIP Secure (SIPS) protocol [54] is a standard protocol for securing SIP based on TLS. It constructs TLS secure tunnels among each hop in the path between end points.

TLS adopts PKI to ensure confidentiality of all transmitted data, and to verify and authenticate the validity of each party in the context of public and private keys. Unfortunately, PKI is vulnerable to MITM attacks [55,56]. In addition, maintaining many open TLS connections simultaneously may significantly reduce the performance of SIP servers, rendering them unable to process incoming requests due to resource exhaustion, making servers more vulnerable to denial-of-service attacks [8,10,57,58]. Moreover, as we prove in Section 4, the standard password-based authentication protocol (SRP) supported by TLS (SRP-TLS) [26] is vulnerable to offline password guessing attacks, stolen verifier attacks, and Denning–Sacco attacks. The proposed protocol (S-SIP) covers security flaws and weaknesses in SIPS. S-SIP does not depend on TLS. However, it is a redesign for SIP taking into account its security requirements. In addition, the performance of the proposed protocol is better than SIPS, as explained in Section 8.

(3) S/MIME

S/MIME [14] is a standard defined in RFC 3851. Multipurpose internet mail extensions (MIME) and its security improvement S/MIME were originally designed for e-mail services. Additionally, they are used by other text-based internet application protocols. S/MIME is the de facto standard for securing emails by using a PKI infrastructure and X.509 certificates. S/MIME supports the exchange of complex messages along with preserving a set of security objectives, including confidentiality, integrity, and authenticity. S/MIME provides end-to-end confidentiality for SIP messages, integrity protection for the body and identity authentication for the sender of the message.

S/MIME has many limitations and drawbacks [57,59]. The complexity required to implement S/MIME to protect SIP signaling messages can be a significant factor in limiting its implementation in most environments. In addition, S/MIME encapsulates SIP messages into the MIME body, which creates a considerable overhead and processing cost over SIP messages. Moreover, due to the complexity of managing and distributing security certificates, most commercial SIP clients do not support S/MIME.

(4) IPSec

IPSec [12] is an independent and general purpose network-level protocol in the internet architecture designed by the Internet Engineering Task Force (IETF), which provides protection to IP packets. Thus, protocols such as SIP and RTP can run over it without any changes. IPSec can be used in a tunnel or transport mode to protect its payload. It can provide confidentiality, integrity, and authentication services for media and SIP signal messages by creating secure tunnels between end points. However, it has many drawbacks [8–10,57]:

- Since IPSec is implemented at the operating system level or kernel layer, many SIP clients do not support it. Thus, IPSec can be utilized to protect signaling traffic between SIP servers but not between servers and clients.
- For applications working on top of IPSec, such as SIP, it is difficult to detect whether IPSec has failed to be set up. As a result, there are advantages for security mechanisms working at layers above the IP layer.
- Deploying IPSec in VoIP systems requires more effort because of its complexity and infrastructure requirements compared to other protocols.
- It does not scale well for large, distributed networks and distributed applications.

2.2. Research-Based Solutions

Numerous security schemes have been introduced in the literature for securing SIP. However, most of them are only concerned with SIP authentication. The following discusses the most significant SIP authentication schemes.

In [29], Yang et al. proposed an authentication scheme based on the Diffie–Hellman key exchange protocol. Later, Huang et al. [30] identified that Yang et al.'s protocol was insecure against offline password guessing attacks. Therefore, they presented an improved scheme. Later, it was demonstrated that Huang et al.'s scheme could not thwart offline password guessing attacks [31].

Based on Yang et al.'s study, in [32,33] authentication schemes for SIP were proposed based on ECC [34]. However, Yoon et al. [35] showed that these schemes were susceptible to offline password guessing, Denning–Sacco, and stolen verifier attacks. To overcome these weaknesses, Yoon et al. proposed an enhanced authentication scheme for SIP with more security. Unfortunately, Pu [36] showed that the scheme of Yoon et al. was still prone to offline password guessing and replay attacks.

To reduce the high computational cost, Tsai [37] suggested an efficient authenticated key agreement scheme adopting only one-way hash functions and exclusive-OR operations. Nevertheless, in [38] it was proven that Tsai's scheme could not provide perfect forward secrecy and was vulnerable to offline password guessing attacks, Denning–Sacco attacks, and stolen-verifier attacks. Thus, Yoon et al. [38] proposed an enhanced scheme to overcome the shortcomings of Tsai's scheme. The proposed scheme was based on the elliptic curve discrete logarithm problem (ECDLP). However, Xie [39] demonstrated that Yoon et al.'s scheme could not resist offline password guessing and stolen-verifier attacks. Then, he proposed an improved scheme to overcome the weaknesses of the scheme introduced in [38]. Nevertheless, Farash and Attari [40] demonstrated that the scheme introduced in [39] was still insecure and vulnerable to offline password guessing and impersonation attacks. To avoid these vulnerabilities in Xie's scheme, they presented an improved scheme. Based on the work introduced in [40], Zhang et al. [41] proposed an authentication scheme for SIP.

Lu et al. [42] found that Zhang et al.'s scheme was insecure against insider attacks and could not provide proper security. To overcome the weaknesses of Zhang et al.'s scheme, Lu et al. proposed an improved authentication scheme. They demonstrated that their scheme was resistant to possible known attacks. In [43] the authors stated that Lu et al.'s scheme was prone to user and server impersonation attacks. Therefore, they proposed an enhanced scheme that resists these attacks. Nevertheless, the authors in [44] found that the work introduced in [43] was still vulnerable to the attacks that appeared in Lu et al.'s scheme. Meanwhile, they indicated that Lu et al.'s [42] scheme was insecure against impersonation and identity guessing attacks.

In [45], to overcome vulnerabilities in previous schemes, Zhang et al. developed an authentication and key agreement scheme for SIP using a smart card. In addition to the password, the smart card is used as a second authentication factor. The authors showed that their scheme was efficient and secure against several attacks. Irshad et al. [46] proved that Zhang et al.'s scheme could suffer from impersonation attacks. Consequently, they suggested an improvement for Zhang et al.'s authentication scheme. However, Arshad and Nikooghadam [47] proved that the scheme introduced in [46] could not prevent user impersonation attacks.

In [48], Tu et al. developed an authentication scheme that had low computational costs, based on Zhang's scheme [45]. Chaudhry et al. [49] showed that Tu et al.'s scheme [48] was not immune to server impersonation attacks and replay attacks. Thus, they developed a lightweight authentication and key agreement protocol for SIP. However, Nikooghadam et al. [50] performed cryptanalysis on Chaudhry et al.'s scheme and showed its weakness against password guessing attacks. They developed an enhanced scheme to overcome this weakness. Later, Ravanbakhsh et al. [51] demonstrated that the work introduced in [50] was insecure because it did not provide perfect forward secrecy. Thus, they presented a two-factor authentication and key agreement scheme for SIP networks and showed that it was resistant against various active and passive attacks. Nevertheless, it was proven that Ravanbakhsh et al.'s scheme [51] did not provide perfect forward secrecy [52].

Few studies have been introduced in the literature that provide confidentiality, integrity, and authenticity services for VoIP systems. The most recent and significant studies are discussed in the following. In [60], the authors studied security threats and attacks in internet protocol multimedia subsystem (IMS) networks. In addition, they presented a framework model for protecting messages exchanged between users and for user authentication using IPsec and HTTP digest protocols on every single gateway router. HTTP digest provides user authentication and a limited degree of integrity protection. The IPsec protocol protects confidentiality and integrity at the network layer. However, as explained in Section 2.1, HTTP digest and IPsec protocols have many drawbacks.

In [61], Farley et al. presented a system called VoIP Shield to mitigate MITM and replay attacks. The VoIP Shield consists of two shields with a pre-distributed shared key; one of the shields is connected to the client and the other is connected to the proxy server to protect VoIP traffic exchanged between them. For message authentication, the shields use a simple hash-based message authentication code scheme. However, the VoIP Shield did not support any scheme for entity authentication or key distribution and management which made it vulnerable to many VoIP attacks.

Basem et al. [62] provided a multilayered security solution based on different open-source applications: Snort, IPtables, SnortSam, and OpenVPN tunnel. The design provided reliable and secure VoIP services for users to prevent denial-of-service and eavesdropping attacks. Signaling messages were protected using the OpenVPN tool which transmits traffic over TLS-based VPNs. The Snort tool was used for detecting security violations and attacks. However, as explained in Section 2.1, the TLS protocol has many drawbacks. In addition, intrusion detection and prevention tools or techniques affect the system's performance.

3. Background

3.1. Session Initiation Protocol

SIP has been standardized by the IETF [2]. It is an application layer signaling protocol for setting up, modifying and terminating multimedia IP sessions, including VoIP telephony, video, streaming media, and instant messaging. SIP is a text-based protocol based on the HTTP protocol, which defines two types of messages: SIP requests and responses. The SIP protocol follows the client/server model whose basic components are [9]:

(1) User Agent Client and Server:

An SIP user agent is a logical network endpoint used to create or receive SIP messages and thereby manage an SIP session. The user agent itself has a client element, called the user agent client (UAC), and a server element, called the user agent server (UAS). The UAC is responsible for creating requests and the UAS processes and responds to each request generated by a UAC. The SIP user agent (UA) can be lightweight clients suitable for embedding into end-user devices such as mobile handsets; alternatively, they can be desktop applications (e.g., softphones).

(2) Registrar Server

The SIP registrar server is responsible for user registration. It has a database containing the location and user's preferences as indicated by the user agents. The registrar server accepts SIP registration requests and binds the information it receives (the SIP address and associated IP address of the registering device).

(3) Proxy Server

The proxy server is an intermediary entity that acts as both a server and a client for the purpose of making requests on behalf of other clients. A proxy server primarily plays the role of routing. Its job is to ensure that a request is sent to another entity (proxy server) closer to the targeted user agent. Users in an SIP environment are identified by SIP uniform resource identifiers (URIs). The format of an SIP URI is similar to an e-mail address, generally consisting of a username and a domain name.

SIP is designed to be simple and easy to use. It defines six main types of messages, namely INVITE, ACK, BYE, CANCEL, OPTIONS, and REGISTER. Figure 1 shows an

example of a call flow from one UA (UA1) to another (UA2) [2]. A session is initiated when UA1 sends an INVITE message to the appropriate proxy server indicating that UA1 wishes to communicate/talk with UA2 [9]. Immediately, the proxy server sends a response message (TRYING) to UA1 to acknowledge that the INVITE message is handled and attempts to resolve the called user's location and sends the request to UA2. UA2 sends a response (RINGING) when their telephone begins to ring. Finally, when UA2 receives the request and picks the call, it sends the OK message to the proxy server that forwards it to UA1. Upon receiving the OK message, UA1 sends the ACK message to UA2. Upon reception of the ACK message, UA2 establishes the connection with UA1. Then, media streams are exchanged directly between them.

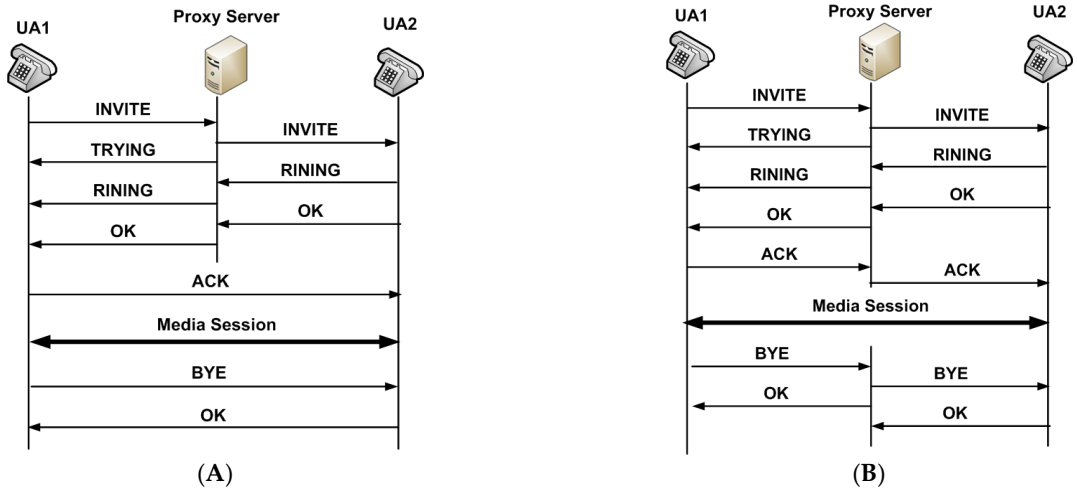

Figure 1. An example of a SIP call: (**A**) without the record-route option; (**B**) with the record-route option.

The session ends with a BYE request message which is routed directly from UA1 to UA2 that sends a reply with an OK message, as shown in Figure 1A. In some cases, it may be useful for proxy servers in the SIP signaling path to see all messages exchanged between the endpoints for the duration of the session. In these cases, the proxy servers insert a record-route field in the INVITE and OK messages, which enforces the BYE and OK messages to be routed through the proxy servers [2], as shown in Figure 1B. If there are more than one proxy server between UA1 and UA2, all SIP messages received from user agents are routed through the proxy servers.

3.2. Secure Remote Password Protocol

The SRP protocol is an A-PAKE protocol proposed in [24]. It is also known as SRP-3 and described in RFC 2945 [25]. SRP-6 [28] is a variant of SRP-3 that is included in IEEE P1363 [27] and ISO/IEC 11770-4 standards. Additionally, it is described in RFC 5054 for strong password authentication in SSL/TLS [27]. The SRP protocol is currently at revision 6a [17,28].

SRP is suitable for secure password verification and session key generation over an insecure communication channel. The protocol allows the participants to establish secure sessions without actual exchange of a password or any other information that can be derived from the password. Furthermore, being an augmented PAKE protocol, the server does not store any password-equivalent data. Therefore, the SRP protocol allows a client and a server to authenticate each other without transmitting a password or trusting a third party.

The main goal of SRP is to establish a key agreement between two parties in a client/server model to authenticate themselves in a manner similar to Diffie–Hellman key exchange. Figure 2 shows the operation of the SRP-6a protocol between client *C* and server *S*. Additionally, Table 1 shows the notation used in this section and Sections 4 and 5.

In SRP, all computations are performed in a finite Galois Field $GF(p)$ defined by a large prime p and a base element g that generates a large multiplicative subgroup of order q. The prime number p must be large enough so that computing discrete logarithms modulo p is infeasible. In addition, all computations are performed modulo p. This means that g^x should be read as $g^x \ mod \ p$.

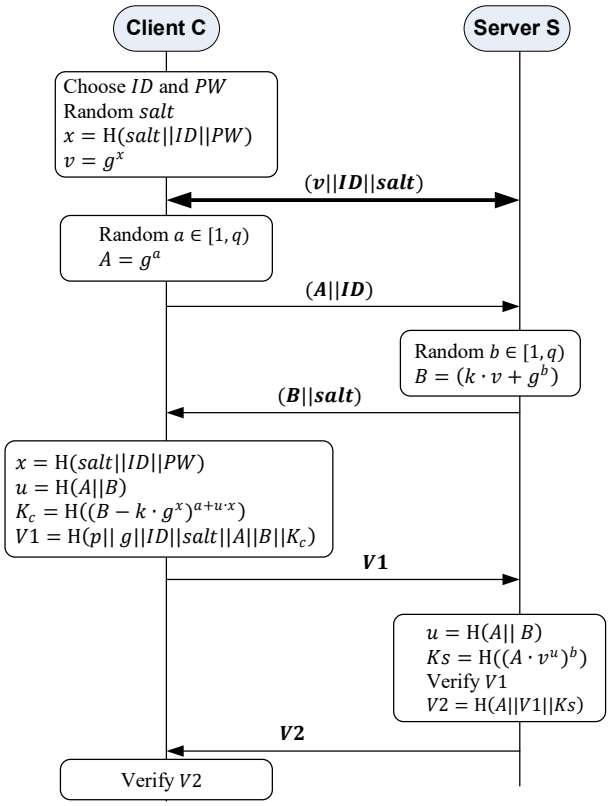

Figure 2. The secure remote password protocol SRP-6a.

Table 1. Notations.

Notation	Description
p	Large prime number
g	Primitive root modulo p (generator)
$\mathrm{H}(\cdot)$	One-way hash function
PW_C	Password of client C
K_Y	Session key of entity Y
K_{XY}	Session key shared between X and Y
a, b	High-entropy random numbers
| |	Concatenation operation
TS_i	Timestamp number i
LT_i	Lifetime number i
N_i	Nonce number i
$\oplus$	Bitwise XOR operation
$\leftrightarrow$	Secure channel
$\rightarrow$	Common channel
ID_C	ID of entity C
E(K, [M])	Symmetric encryption of M with the key K

The SRP protocol consists of three phases: registration phase, login phase, and authentication phase. In the registration phase, the user of a client C must register their password with the server S. The user freely chooses their identity ID and a password PW. Then, C generates a random password $salt$ and computes a verifier v and a secret exponent x,

as shown in Figure 2. Then, C sends v, ID, and $salt$ to S through a pre-established secure channel. S stores v and $salt$ in its database indexed by ID.

In the login phase, the user tries to login into S over an insecure channel. C generates a random number $a \in [1,\ q)$ and computes the ephemeral public key $A = g^a$. Then, C sends ID and A to S. Upon receiving the message, using the client's ID, S retrieves the corresponding $salt$ and the verifier v. Then, it generates a random number $b \in [1,\ q)$ and computes the ephemeral public key $B = k{\cdot}v + g^b$. Next, it sends back B and $salt$ to C. Upon receiving the message, C computes x, u, K_c, and $V1$, as shown in Figure 2. u is a parameter that two parties can compute using the public outputs: A and B. K_c is the client key that is simplified as:

$$K_c = \mathrm{H}\Big((B - k{\cdot}g^x)^{a+u\cdot x}\Big) = \mathrm{H}\Big(\Big(k{\cdot}v + g^b - k{\cdot}g^x\Big)^{a+u\cdot x}\Big) = \mathrm{H}\Big(g^{b(a+u\cdot x)}\Big)$$

Additionally, S can compute their session key K_S as:

$$K_S = \mathrm{H}\Big((A{\cdot}v^u)^b\Big) = \mathrm{H}\Big((g^a{\cdot}g^{x\cdot u})^b\Big) = \mathrm{H}\Big(g^{b(a+u\cdot x)}\Big)$$

Thus, at the end of the second phase, the client and server establish a common session key $K_{SC} = K_S = K_c$.

In the last phase, the authentication phase, the two parties prove to each other that their keys are identical using two verification massages, $V1$ and $V2$. First, C computes $V1$ and sends it to S, as shown in Figure 2. S receives $V1$ and computes $V1^*$ as:

$$V1^* = \mathrm{H}(p||g||ID||salt||A||B||\ K_S)$$

S checks if $V1 = V1^*$. If so, this means that $K_s = K_c$. Consequently, S computes $V2$ and sends it to C, as shown in Figure 2. After receiving $V2$, C computes $V2^* = \mathrm{H}(A||V1||K_C)$. If $V2$ and $V2^*$ match, the session key is verified and S and C authenticate each other. If the verification of $V1$ or $V2$ fails, either C or S terminates the authentication process.

4. Security Analysis of SRP

The SRP-6a protocol does not have a proof model for security. Recently, a formal analysis of SRP was introduced in [63]. The authors analyzed SRP-3 using a cryptographic protocol shape analyzer (CPSA). They found that the structure of the protocol did not have any major weaknesses such as leakage of secret keys. In addition, they indicated that SRP could not resist malicious server attacks if the verifier v and the random number b were revealed (not only v). However, they did not explain how to perform this attack. In this section, we analyze SRP-6a informally. We found that it is vulnerable to offline password guessing, Denning–Sacco, and stolen-verifier attacks, as explained below in detail.

4.1. Offline Password Guessing Attack

In remote user authentication schemes, the user is allowed to choose their password. The user tends to choose a password that can be easily remembered for their convenience. Therefore, most passwords have low entropy so they are vulnerable to password guessing attacks. In this attack, an attacker intercepts authentication messages and stores them locally. Then, they attempt to use a guessed password to verify the correctness of their guess using these authentication messages. The SRP protocol is vulnerable to an offline password guessing attack. An attacker can perform this attack for SRP-6a, shown in Figure 2, as follows:

1. Assuming that the attacker masquerades as a fake server and persuades C to make an authentication attempt.
2. C computes the ephemeral public key A and sends it with its ID to the attacker.
3. To capture $salt$ of C, the attacker starts an authentication session with the actual server S by sending A and ID, received from C, to S.

4. S computes B and sends it to the attacker with the *salt* of client C.
5. The attacker generates a random number b, guesses any password PW^*, and computes their own parameter $x^* = H(salt||ID||PW^*)$ and exponential residue $B^* = k \cdot g^{x^*} + g^b$. Then, they send B^* and $salt$ to the client C.
6. The client C computes the message $V_1 = H(p||g||ID||salt||A||B^*||K_c)$ and sends it to the attacker, where $K_c = H\left((B^* - K \cdot g^x)^{a+u^* \cdot x}\right)$ and $u^* = H(A||B^*)$.
7. The attacker imitates network failure or informs C that the password is not correct.
8. The long-term private password PW can be guessed by performing the following offline password guessing attack:
   (i) The attacker computes $x^* = H(salt||ID||PW^*)$, $B^* = k \cdot g^{x^*} + g^b$, $u^* = H(A||B^*)$, and $v = g^{x^*}$.
   (ii) The attacker computes a modified server key as $K_S^* = H\left(\left(A \cdot v^{u^*}\right)^b\right) = \left(\left(A \cdot g^{x^* \cdot u^*}\right)^b\right)$.
   (iii) The attacker computes a modified message $V_1^* = (p||g||ID||salt||A||B^*||K_S^*)$ and checks if $V_1 = V_1^*$.
   (iv) If it holds, the attacker has guessed the correct secret password $PW^* = PW$.
   (v) If it is not correct, the attacker chooses another password from a dictionary and repeatedly performs the above verification process starting from step i.

Compromising the user's secret password PW enables the attacker to impersonate the client or the server.

4.2. Stolen-Verifier Attack

In most existing password-based authentication schemes, servers are always the prime targets of adversaries because the users' verifiers (e.g., passwords) are stored in the server's database. In a stolen-verifier attack [64], an adversary who steals a password verifier from the server can use it to impersonate a legitimate user. In fact, an adversary who obtains a password verifier may further mount a password guessing attack.

In the SRP protocol, if the password verifier v, ID and $salt$ of a user stored in the server have been eavesdropped, the attacker can use v to masquerade as the original server S and can obtain sensitive user information. Additionally, they can easily perform offline password guessing attacks to extract the user's password, which is used to impersonate the client. An offline password guessing attack can be performed as follows:

1. The attacker chooses a secret password PW^* from the password dictionary.
2. The attacker computes $x^* = H(salt||ID||PW^*)$ and $v^* = g^{x^*}$ and checks if $v^* = v$.
3. If it holds, the attacker has guessed the correct secret password $PW^* = PW$.
4. If it is not correct, the attacker chooses another password from the password dictionary and repeatedly performs the above verification process.

Compromising the user's secret password PW enables the attacker to impersonate the client or the server. In addition, using the stolen password verifier v of client C, the attacker executes the server spoofing attack as follows:

1. An active adversary AD may eavesdrop the communication flows between C and S or persuade C to make an authentication attempt with his server.
2. When legal client C wants to login into server S, they send the request message $(A||ID)$ to AD.
3. AD generates a random number b and computes the ephemeral public key $B = k \cdot v + g^b$ using the stolen-verifier v for C.
4. AD sends B along with $salt$ to C.
5. C computes his key $K_c = g^{a \cdot b + b \cdot u \cdot x}$ and the verification message $V1$, as explained in Section 3.2. Then, C sends $V1$ to AD.

6. Upon receiving $V1$ from C, AD computes $u = H(A||B)$ and the verification message $V2 = H(A||V1||K_A)$, where $K_A = H\left((A \cdot v^u)^b\right) = g^{a \cdot b + b \cdot u \cdot x} = K_c$, and sends it back to C as evidence that they have the correct session key.
7. C verifies $V2$ as explained in Section 3.2. Because $K_A = K_c$, C accepts $V2$ and trusts AD as a server. Therefore, AD can obtain the client's personal information.

As explained above, using the stolen user's password verifier, the adversary can impersonate the legal server and can easily find the user's secret password PW using a password dictionary attack. Therefore, the SRP-6a protocol is insecure against stolen-verifier attacks.

4.3. Denning–Sacco Attack

The Denning–Sacco attack [41] occurs when an intruder obtains the session key from an eavesdropped session and uses it either to gain the ability to impersonate the user directly or to conduct a brute-force search against a long-term private key, such as the user's password. This attack arises from the fact that compromising a fresh session key using an old key enables the protocol to be compromised.

In the SRP protocol, if the attacker can leak the session key K_C from the client C, the following Denning–Sacco attack is possible:

1. Assume the attacker intercepts the request message $(A||ID)$ sent by C to S.
2. To capture $salt$, the attacker starts an authentication session with the actual server S by forwarding parameters A and ID to S. S computes B and sends it to the attacker with the $salt$ of client C.
3. The attacker can masquerade as a fake server, randomly generate a number b, guess any password PW^*, and compute his own parameters $x^* = H(salt,\ ID, PW^*)$ and $B^* = k \cdot g^{x^*} + g^b$. Then, they send B^* and $salt$ to C.
4. C computes the message $V1 = H(p,\ g, ID,\ slat,\ A,\ B^*,\ K_C)$ and sends it to the attacker, where $K_C = H\left((B^* - K \cdot g^x)^{a + u^* \cdot x}\right)$ and $u^* = H(A, B^*)$.
5. Assuming that the attacker somehow obtained the shared session key K_{SC} from the client, password PW can be obtained by performing the following offline password guessing attack:
   (i) The attacker makes a guess for the secret password PW^* from a password dictionary.
   (ii) The attacker computes $x^* = H(salt, ID, PW^*)$, $u^* = H(A,\ B^*)$ and $K_S^* = H\left(\left(A \cdot g^{x^* \cdot u^*}\right)^b\right)$ and checks if $K_{SC} = K_S^*$.
   (iii) If it holds, the attacker has guessed the correct secret password $PW^* = PW$. Otherwise, the attacker repeatedly performs the verification process.

Finally, the attacker can obtain the user's secret password, which can be used to impersonate the client or the server.

5. Secure SIP

To secure SIP, this work introduces a new scheme, called secure-SIP (S-SIP). S-SIP consists of two protocols: the SIP authentication protocol (A-SIP) and the key management and protection protocol (KP-SIP). S-SIP provides SIP with essential security services to protect SIP messages against many attacks, which are authentication, integrity, confidentiality, and key management. The authentication service is provided to SIP using the A-SIP protocol, whereas the integrity, confidentiality, and key management services are provided using the KP-SIP protocol. The A-SIP protocol is an authentication protocol that provides mutual authentication for all entities in the VoIP system. KP-SIP is used to exchange session keys and authentication tickets between entities and to secure SIP signal messages exchanged between entities. The following explains these two protocols in detail.

5.1. A-SIP Protocol

The A-SIP protocol is a novel mutual authentication scheme for the session initiation protocol. It is a client/server protocol based on the SRP protocol. However, it overcomes the flaws in the SRP protocol explained in Section 4. The A-SIP protocol is responsible for authenticating entities before the actual communication takes place between them. Figure 3 shows the proposed authentication protocol A-SIP. The used notations are listed in Table 1. The A-SIP protocol uses a four-way challenge/response handshake technique to provide mutual authentication. It consists of four phases: the setup phase, the login phase, the authentication phase and the registration phase. This section explains these phases in detail.

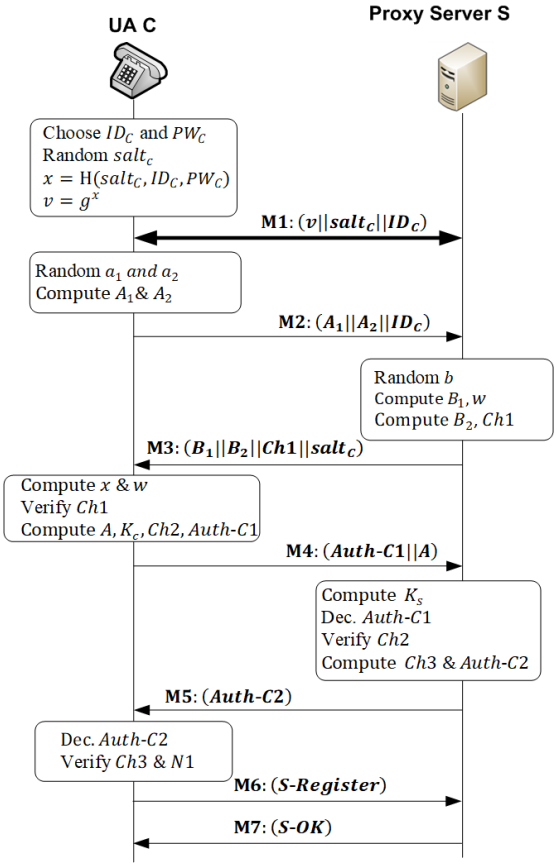

Figure 3. A-SIP protocol.

(1) Setup Phase

In this phase, the user is registered to the remote server S as a legal user by executing the following steps over a secure channel.

1. The user of a client C freely chooses their identity ID_C and password PW_C. Then, C selects a random password $salt_C$ and then computes a verifier v and a secret exponent x as:
$$x = H(salt_C||ID_C||PW_C)$$
$$v = g^x$$
2. C establishes a secure connection with S and sends v, ID_C, and $salt_C$ to S.
3. Upon receiving the message $M1 = (v||salt_C||ID_C)$, S computes the password verifier PW_V as $PW_V = v \oplus H(ID_C||S_P)$, where S_P is the private secret for the server, which is a random bit-string with high entropy. Finally, S stores PW_V and $salt_C$ in its database indexed by ID_C.

(2) Login Phase

In the login phase, the user tries to login into the proxy server S to access different services over an insecure channel. As shown in Figure 3, the steps in this phase (messages $M2$ and $M3$) are executed as follows:

1. C generates two random numbers a_1 and a_2 from the range $[1,\ q)$. Additionally, it computes two ephemeral public keys $A_1 = g^{a_1}$ and $A_2 = g^{a_2}$.
2. C sends the request message $M2 = (A_1||A_2||ID_C)$ to the proxy server.
3. When S receives $M2$, it uses ID_C to retrieve $salt_C$ and PW_V. Using S_P, S computes $H(ID_C||S_P)$, which is XORed with PW_V to obtain v. Then, it generates a random number $b \in [1,\ q)$ and computes the ephemeral public key $B_1 = g^b$ and private key $w = H\left(v^b||A_1||A_2\middle||salt_C\right)$. If S does not have credentials of C, it securely communicates with the authentication/registrar server to get them.
4. S computes the second ephemeral public key B_2 and the first challenge $Ch1$ as:

$$B_2 = (A_1 \cdot A_2 \cdot v)^{b \cdot w}$$

$$Ch1 = H(w||B_1||B_2) \tag{1}$$

5. S sends the message $M3 = (B_1||B_2||Ch1||salt_C)$ to C.

(3) Authentication Phase

In the authentication phase, based on pre-shared parameters in the preceding phases, S and C verify the identity of each other. If they succeed, they generate and share a unique session key. Figure 3 shows the authentication phase between C and S (messages $M4$ and $M5$). The authentication procedure is performed on a common channel in the following steps:

1. Upon receiving the message $M3$, C computes x and w as:

$$x = H(salt_C||ID_C||PW_C)$$
$$w = H(B_1^x||A_1||A_2||salt_C) = H\left(g^{x \cdot b}||A_1||A_2\middle||salt_C\right) = H\left(v^b||A_1||A_2\middle||salt_C\right)$$

Then, it computes the first challenge $Ch1$ using Equation (1).

2. C checks that the received challenge $Ch1$ is equal to the computed challenge. If not true, C terminates the session with S and does not complete the authentication process. Otherwise, C starts to compute the second challenge.
3. C computes the public key A, the session key K_C, and the second challenge $Ch2$ as:

$$A = (A_1 \cdot B_1 \cdot v)^{a_2 \cdot w}$$
$$K_C = \left(\frac{B_2}{B_1^{a_2 \cdot w}}\right)^{a_2} = \left(\frac{(A_1 \cdot A_2 \cdot v)^{b \cdot w}}{B_1^{a_2 \cdot w}}\right)^{a_2} = \left(\frac{(A_1 \cdot g^{a_2} \cdot v)^{b \cdot w}}{g^{b \cdot a_2 \cdot w}}\right)^{a_2} =$$
$$= (A_1 \cdot v)^{a_2 \cdot b \cdot w} = g^{(a_1 + x) a_2 \cdot b \cdot w} \tag{2}$$
$$Ch2 = H(\mathrm{H}(p||g||salt_C) \oplus H(ID_C||B_1) \oplus H(A||B_2||w))$$

4. C generates a nonce $N1$, which is a random number that is difficult for an opponent to guess and represents a unique identifier for this transaction. C encrypts $N1$ and $Ch2$ using K_C to construct the first authenticator $Auth\text{-}C1 = E(K_C,\ [Ch2||N1])$ that represents the client authenticator. The subsequent message (response) received from S must contain the hash of the nonce $N1$.
5. C sends the challenge message $M4 = (Auth\text{-}C1||A)$ to S
6. When S receives the message, it computes its session key K_S as:

$$K_S = \left(\frac{A}{A_2^{b \cdot w}}\right)^b = \left(\frac{(A_1 \cdot B_1 \cdot v)^{a_2 \cdot w}}{A_2^{b \cdot w}}\right)^b = \left(\frac{\left(A_1 \cdot g^b \cdot v\right)^{a_2 \cdot w}}{g^{a_2 \cdot b \cdot w}}\right)^b$$

$$= (A_1 \cdot v)^{a_2 \cdot b \cdot w} = g^{(a_1+x)a_2 \cdot b \cdot w} \quad (3)$$

From Equations (2) and (3), it is clear that $K_S = K_C = K_{SC}$, where K_{SC} is the shared session key between S and C. Next, S decrypts $Auth$-C1 using K_S to extract $Ch2$ and $N1$. Then, it computes the second challenge $Ch2^*$.

7. S compares the second challenge $Ch2$ extracted from $Auth$-C1 and the computed challenge $Ch2^*$. If $Ch2 \neq Ch2^*$, S aborts the session with C. Otherwise, S computes the third challenge $Ch3 = H(H(A||B_2) \oplus H(Ch2||salt_C))$
8. S encrypts $Ch3$ and $H(N1)$ using session K_S to construct the server authenticator as:

$$Auth\text{-C2} = E(K_S, [Ch3||H(N1)])$$

9. After receiving the message $M5 = Auth$-C2, C decrypts the message using K_C to extract $Ch3$. Next, it verifies $Ch3$ and $H(N1)$. If not correct, C terminates the sessions. Otherwise, C assures that the message has been sent by S and the authentication is successful. Using the nonce $N1$ assures C that this is a response for a fresh message and helps prevent a replay attack.

(4) Registration Phase

This is the last phase in the A-SIP protocol. In this phase, the client registers its URI address with the Proxy/Registrar SIP server at which the user can be reached. As explained in Section 3, registration with a local server is essential to receive or make an SIP call. The steps of this phase (messages $M6$ and $M7$) are as follows:

1. At the end of the authentication phase, C forms the *S-Register* message as

$$S\text{-}Register = E\Big(K_{SC}, \Big[Register\Big|\Big|H^2(N1)\Big]\Big)$$

where *Register* is the standard SIP registration message and $H^2(N1) = H(H(N1))$.
2. On receipt of the *S-Register* message, S decrypts the message to obtain the *Register* message and checks $H^2(N1)$. The SIP server processes the *Register* message to register the URI of the client C (URI_c).
3. S sends the *S-OK* message to C, which is computed as:

$$S\text{-}OK = E(K_{SC}, [OK||H^3(N1)||TKT_{CS}])$$
$$TKT_{CS} = E(K_P, [URI_C||IP_S||TS_1||LT_1])$$

where OK is the standard SIP OK message, $H^3(N1) = H\big(H^2(N1)\big)$, and TKT_{CS} is a ticket that C uses for subsequent authentications to obtain VoIP services from S. This is explained in detail in the next section.
4. When receiving the *S-OK* message from S, C decrypts it using the session key and checks $H^3(N1)$. If correct, C accepts the authentication ticket TKT_{CS}.

The ticket TKT_{CS} contains the registered address of the client URI_C and the network address of the proxy server IP_S. To prevent the adversary from reusing the ticket, it includes a timestamp TS_1, indicating the date and time at which the ticket was issued, and a lifetime LT_1, indicating the length of time for which the ticket is valid. In addition, the ticket is encrypted using the private key of the server K_P. Thus, it cannot be modified by C or by an opponent. The ticket helps minimize the number of times that a user has to enter a password. The ticket is reused for a single login session. For the lifetime of the ticket, C can use the ticket for multiple accesses to the same proxy server, as explained in the next section. If C moved to another region managed by another SIP server, it must register to the new SIP server to obtain a new authentication ticket.

5.2. KP-SIP Protocol

The KP-SIP protocol is used for key management between entities, and for protecting SIP messages exchanged between different entities. Figure 4 explains the session example using KP-SIP when a single proxy is involved. Tables 2 and 3 show the details of the exchanged messages shown in Figure 4. The KP-SIP protocol is divided into two phases: the call initiation phase (messages M1 to M10) and the call teardown phase (messages M11 and M12). Next, these phases are explained.

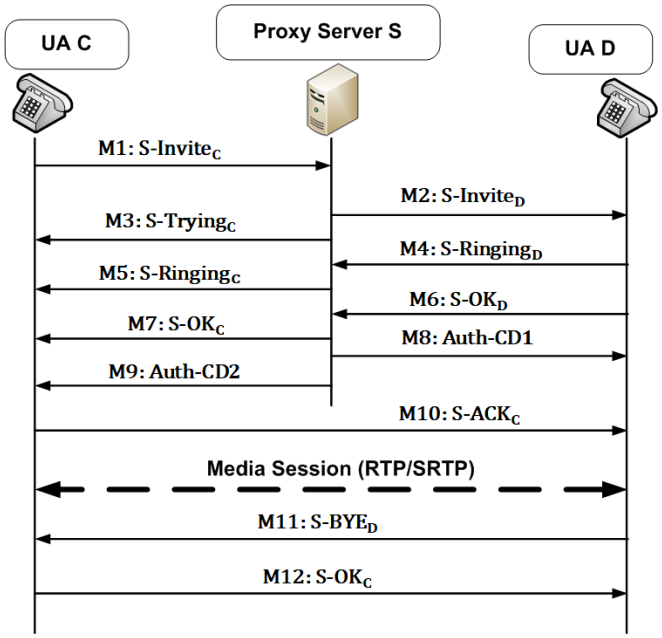

Figure 4. KP-SIP protocol.

Table 2. Messages M1 to M7 of the KP-SIP protocol.

(M1) S-INVITE$_C$ = $E(K_{SC}, [\text{INVITE}_c \| Auth\text{-}C3 \| TKT_{CS} \| TS_2 \| LT_2])$
$Auth\text{-}C3 = (URI_C \| IP_S \| N2)$
(M2) S-INVITE$_D$ = $E(K_{SD}, [\text{INVITE}_D \| N3 \| TS_3 \| LT_3])$
(M3) S-TRYING$_C$ = $E(K_{SC}, [\text{TRYING}_C \| H(N2)])$
(M4) S-RINGING$_D$ = $E(K_{SD}, [\text{RINGING}_D \| H(N3)])$
(M5) S-RINGING$_C$ = $E(K_{SC}, [\text{RINGING}_C \| H^2(N2)])$
(M6) S-OK$_D$ = $E(K_{SD}, [\text{OK}_D \| H^2(N3)])$
(M7) S-OK$_C$ = $E(K_{SC}, [\text{OK}_C \| H^3(N2)])$

Table 3. Messages M8 to M12 of the KP-SIP protocol.

(M8) $Auth\text{-}CD2 = E(K_{SC}, [K_{CD1} \| URI_C \| URI_D \| IP_S \| H^4(N2) \| N4 \| TS_4 \| LT_4])$
(M9) $Auth\text{-}CD1 = E(K_{SD}, [K_{CD1} \| URI_D \| URI_C \| IP_S \| H^3(N3) \| \text{N4} \| TS_5 \| LT_5])$
(M10) S-ACK$_C$ = $E(K_{CD1}, [\text{ACK} \| K_{CD2} \| H(N4)])$
(M11) S-BYE = $(K_{CD2}, [\text{BYE} \| H^2(\text{N4}) \| TS_6 \| LT_6])$
(M12) S-OK = $(K_{CD2}, [\text{OK} \| H^3(\text{N4})])$

(1) Call Initiation Phase

Let user agent *C* attempt to call user agent *D*, where *C* and *D* are registered with proxy server *S*. Figure 4 shows the call flow. *C* initiates the session by sending the S-INVITE$_C$ message to the proxy server. As shown in Table 2, S-INVITE$_C$ contains the standard SIP invite message INVITE$_C$ and the ticket obtained from the proxy server in the registration phase, the authenticator $Auth\text{-}C3$, the timestamp and the message lifetime. The message

S-INVITE$_C$ is encrypted using the last shared session key K_{SC} with S. The authenticator $Auth$-$C3$ includes the URI of C, the network address IP_s of S, and the nonce $N2$.

After receiving the S-INVITE$_C$ message, S decrypts it using the last shared session key with C. Then, it checks TS_2 and LT_2 to ensure that the lifetime of the message has not expired. Then, it decrypts the ticket using its private key and verifies its validity using the timestamp and the lifetime included in the ticket. In addition, it compares URI_C and IP_S included in authenticator $Auth$-$C3$ with those included in the ticket. If the ticket is valid, IP_S is correct, and URI_C is correct and matches the registered URI of C, S authenticates C and processes the SIP INVITE message. To prevent a replay attack, the S-INVITE$_C$ message has a very short lifetime compared to the ticket lifetime. Additionally, to prevent an opponent from replaying ticket TKT_{CS}, it is encrypted with the S-INVITE$_C$ message using the session key.

The server sends the S-INVITE$_D$ message to D, encrypted by the session key K_{SD} shared between S and D. The S-INVITE$_D$ message contains its lifetime LT_3, timestamp TS_3, and nonce $N3$ to avoid a replay attack.

The server sends the S-TRYING$_C$ message to C that contains the standard SIP TRYING message and $H(N2)$, which are encrypted using the session key K_{SC}. S returns the hash of $N2$ received in M1 to C to show the freshness of the reply. Every time C sends S-INVITE$_C$, it starts a timer. If C receives S-TRYING$_C$ before the timer expires, the timer is stopped, and the sender sends the next message. If the timer expires or $N2$ is not correct, C terminates the session and initiates another session by sending the S-INVITE$_C$ message.

As shown in Figure 4 and Table 2, messages M4 to M7 consist of the standard SIP RINGING or OK messages and the hash of the last received nonce $\left(H^j(N) = H^{j-1}(N)\right)$. These messages are encrypted using the session keys shared between each user agent and the proxy server to protect them from alteration or spoofing. Additionally, the nonce is used to assure that the messages are fresh and have not been replayed by an adversary. D sends the message S-OK$_D$ if it accepts the call. S forwards the SIP OK message to C using the S-OK$_C$ message.

For mutual authentication between C and D, the server sends the authenticators $Auth$-$CD1$ and $Auth$-$CD2$, as shown in Figure 4. For message integrity and mutual authentication, the contents of $Auth$-$CD1$ and $Auth$-$CD2$ are encrypted using the shared session key between each user agent and the server, as shown in Table 3. $Auth$-$CD1$ and $Auth$-$CD2$ contain a copy of the session key K_{CD1}, which is used for protecting messages exchanged directly between C and D. Additionally, they have several pieces of information: URI of C, the network address of the server IP_S, the hash of the last nonce exchanged with the server, the nonce $N4$, the timestamp, and the lifetime of the message. As explained above, these pieces are included to prevent spoofing and replaying attacks.

At the end of the call initiation phase, C sends the S-ACK$_C$ message to D. The contents of this message are encrypted using the shared session key K_{CD1} sent by S in the authenticators $Auth$-$CD1$ and $Auth$-$CD2$ in messages M8 and M9. The S-ACK$_C$ message contains the standard SIP ACK message, the hash of $N4$ proposed by S in the authenticators. Additionally, it contains a new random session key K_{CD2} suggested by C. Encrypting the content of the S-ACK$_C$ message using K_{CD1} and exchanging $N4$ assures mutual authentication and prevents a replay attack. After receiving S-ACK$_C$, D can extract and process the standard SIP ACK message to start a media session with C. Protecting media streams is out of the scope of the proposed work.

(2) Teardown Phase

In this phase, one of the user agents ends the SIP session by sending the S-BYE message. The other user agent responds with the S-OK message. These messages include the standard SIP BYE and OK messages. Additionally, to prevent a replay attack, the S-BYE message includes the hash of the nonce received in the messages M10, the timestamp, and the lifetime. The contents of these messages are encrypted using the last session key K_{CD2} shared between C and D.

If the record-route option is enabled, the BYE and OK messages are routed through proxy servers, as explained in Section 3. In this case, the BYE and OK messages are encrypted using the session key exchanged between the user agents and the server (K_{SC} and K_{SD}). If the messages exchanged between UA1 and UA2 are routed through multiple SIP servers, as shown in Figure 5, we suppose that the connections between SIP servers are protected using symmetric or asymmetric cryptography algorithms. In addition, routing messages through servers owned by different telecoms/providers is out of scope this work. KP-SIP can be extended to protect connections between different SIP servers.

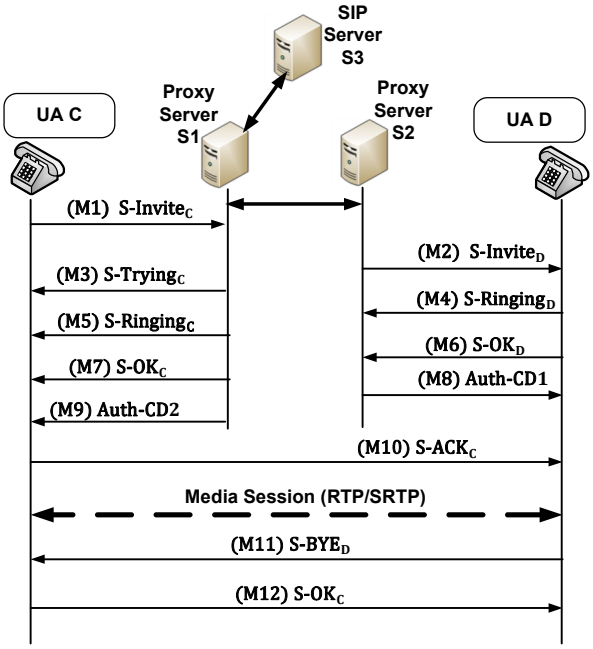

Figure 5. KP-SIP protocol with two proxy servers.

6. Informal Security Analysis

In this section, we prove that the proposed protocol S-SIP is secure and can withstand various attacks and provide security requirements for SIP as follows. Similar to related studies, we assume that the adversary/attacker AD can insert, capture, delete, or modify any messages in the public insecure channel.

6.1. Offline Password Guessing Attack

As explained in Section 4, to perform an offline password guessing attack, an active adversary AD eavesdrops the communication flows between C and S and masquerades as a fake server or client. AD tries to modify and intercept any message that can be exploited to perform the attack. Then, the adversary goes offline and uses a dictionary to test passwords against the intercepted messages. If AD tries to impersonate the server, the attack is performed as follows:

1. During the authentication process, AD intercepts the request message $M2 = (A_1||A_2||ID_C)$ sent by C to login to S.
2. The attacker forwards $M2$ to S to obtain $salt_C$. Then, they generate a random number b, guesses any password PW^*, and computes their own parameter as:

$$x^* = H(salt_C^*||ID_C||PW^*)$$
$$w^* = H\left(g^{x^* \cdot b}||A_1||A_2\Big|\Big|salt_C\right)^{b \cdot w^*}$$
$$B_2^* = \left(A_1 \cdot A_2 \cdot g^{x^*}\right)$$
$$Ch1^* = H(w^*||B_1||B_2^*)$$

Then, they send $M3 = (B_1||B_2^*||Ch1^*||salt_C)$ to C

3. Upon receiving $M3$, C computes x, w, and $Ch1$. Next it compares the received challenge $Ch1^*$ and the computed challenge $Ch1$. If the guessed passwords PW^* are not correct, C detects that $Ch1^* \neq Ch1$. Therefore, C terminates the connection with the fake server.

As shown in the former steps, AD did not receive any message that can be used to check the correctness of the guessed parameters. Therefore, AD cannot perform an offline password guessing attack.

If AD tries to impersonate the client with user identity ID_C, the attack is performed as follows:

1. AD randomly generates a_1 and a_2 and calculates public keys A_1 and A_2 and then transmits them to S with user identity ID_C.
2. S randomly generates a random number b, calculates public keys B_1 and B_2, the secret parameter w, and the challenge $Ch1$. Next, it sends B_1, B_2, $salt_C$, and $Ch1$ to AD.
3. AD guesses any password PW^* and computes other parameters:

$$\begin{gathered} x^* = H(salt_C||ID_C||PW^*) \\ w^* = \mathrm{H}\left(B_1^{x^*}||A_1||A_2\middle||salt_C\right) \\ A^* = \left(A_1 \cdot B_1 \cdot g^{x^*}\right)^{a_2 \cdot w^*} \\ K_c^* = \left(B_2 / B_1^{a_2 \cdot w^*}\right)^{a_2} \\ Ch2^* = \mathrm{H}(\mathrm{H}(p||g||salt_c) \oplus H(ID_C||B_1) \oplus H(A^*||B_2||w^*)) \\ Auth\text{-}C1^* = E(K_c^*,\ [Ch2||N1]) \end{gathered}$$

Then, AD sends the message $M4 = (Auth\text{-}C1^*||A^*)$ to S.

4. Upon receiving $M4$, S encrypts the message using its key $K_s = \left(A^* / A_2^{b \cdot w}\right)^b$. If the guessed password and secret key are wrong, S detects that $Auth\text{-}C1^* \neq Auth\text{-}C1$. Therefore, S does not respond to AD and terminates the session.

As clear in this attack, AD did not obtain any response message from the server that can be used to perform an offline password guessing attack. As a result, impersonation as a legal client or server does not enable the attacker to perform an offline password guessing attack. Therefore, the proposed authentication protocol is not vulnerable to offline password guessing attacks.

6.2. Denning–Sacco Attack

As explained in Section 4, a Denning–Sacco attack refers to obtaining a long-term key such as the user's password or the session key through an obtained old session key. The attacker tries to guess either the user's password or the session key using an old, compromised session key. In the proposed protocol, as shown in Equations (2) and (3), the session key is computed as

$$K_C = K_S = \left(B_2 / B_1^{a_2 \cdot w}\right)^{a_2} = \left(A / A_2^{b \cdot w}\right)^b$$

where $w = \mathrm{H}\left(B_1^x||g^{a_1}||g^{a_2}\middle||salt_C\right)$. The random numbers a_1, a_2, and b are changed for every session. Therefore, if a passive eavesdropper acquires an old session key, they are not able to compute the server's or client's session key. In addition, the attacker cannot perform offline password guessing attacks using the old session key due to the difficulty of the computational Diffie–Hellman problem, which is computationally infeasible with large random numbers.

Let the active eavesdropper AD acquire the session key K_{SC} from the client. Let the eavesdropper impersonate server S and perform a Denning–Sacco attack explained in Section 4.3 as follows:

1. During the authentication process, AD intercepts the request message $M2 = (A_1||A_2||ID_C)$ sent from C to S. Then, they forward $M2$ to S to obtain $salt_C$.
2. AD generates a random number b, guesses any password PW^*, and computes their own parameter as:

$$x^* = H(salt_C,\ ID_C, PW^*)$$
$$w^* = H\left(g^{x^* \cdot b}||A_1||A_2\middle||salt_C\right)$$
$$B_2^* = \left(A_1 \cdot A_2 \cdot g^{x^*}\right)^{b \cdot w^*}$$
$$Ch1^* = H(w^*||B_1||B_2^*)$$

   Then, they send $M3 = \left(B_1^*||B_2^*||Ch1^*\right)$ to C.
3. To perform the brute-force attack using the compromised session key K_{SC}, AD must first compute the server session key as $K_S^* = \left(A / A_2^{b \cdot w^*}\right)^b$. To compute K_S^*, AD must obtain A from C.
4. Upon receiving $M3$, C computes x, w, and $Ch1$. Next, it compares the received and computed challenges ($Ch1$ and $Ch1^*$). If the guessed password PW^* is not correct, C detects that $Ch1^* \neq Ch1$. Therefore, C terminates the connection with the server and does not send the parameter A to AD.

In step 4, because the client closed the connection with AD, they could not obtain $A = (A_1 \cdot B_1 \cdot v)^{a_2 \cdot w}$ from C. AD cannot compute A or a_2 using the public key A_2 because they have will face the difficulty of the discrete logarithm problem. Thus, the proposed protocol is not vulnerable to Denning–Sacco attacks.

6.3. Stolen-Verifier Attack

As explained in Section 4, a stolen-verifier attack occurs when an adversary who steals the password-verifier from the server impersonates a legitimate user in the authentication process. Additionally, he may mount a guessing attack to retrieve the user's password.

For the proposed protocol, the server stores ID and $PW_V = v \oplus \mathrm{H}(ID_C||S_P)$ for each client in its database. Let an impersonator be the adversary who steals the password verifier PW_V and ID_C of C from the database. Then, they try to perform an offline password guessing attack. The adversary chooses a secret password PW^* from a password dictionary. Then, they compute $x^* = H(salt_C, ID_C, PW^*)$ and $v^* = g^{x^*}$. Next, they try to guess the private key of the server as S_P^* to compute $PW_V^* = g^{x^*} \oplus \mathrm{H}(ID_C||S_P^*)$. The process is repeated if $PW_V^* \neq PW_V$. This attack is not feasible because S_P has high entropy and cannot be guessed. Therefore, AD cannot retrieve v or x from PW_V. Thus, they cannot compute the correct values for $w^* = \mathrm{H}\left(g^{x^* \cdot b}||A_1||A_2\middle||salt_C\right)$, $B_2^* = (A_1 \cdot A_1 \cdot v^*)^{b \cdot w^*}$ and $Ch1^* = H(w^*||B_1||B_2^*)$. If AD sent an incorrect value of $Ch1^*$, C closes the connection with AD. Therefore, the adversary cannot impersonate the server even if they have the password verifier stored in the server. In addition, they cannot obtain any information from the client to be exploited to perform an offline password guessing attack.

6.4. Perfect Forward Secrecy

Perfect forward secrecy means that if session keys of one or more entities are compromised, the secrecy of old session keys established by the trusted entities are not affected. It also means that a stolen session key does not help an attacker perform a password guessing attack.

In the proposed protocol, as explained in Section 5, the session key K_{SC} is computed using random numbers (private keys) a_1, a_2, and b. These random numbers are changed every session. Therefore, if K_{SC} is compromised, adversary AD cannot obtain the session keys of past sessions. In addition, as explained in Section 6.2, if the session key is compromised, the AD cannot perform offline password guessing attacks. Thus, the proposed protocol has the properties of perfect forward secrecy.

6.5. Impersonation Attack

In impersonation (spoofing) attacks, the adversary tries to masquerade as a legitimate user or server. As explained in Section 6.1 in detail, for the proposed protocol, the adversary cannot perform an offline password guessing attack because they are unable to impersonate either the user or the server. The client or the server closes the connection with the other party if one of the challenges, $Ch1$ or $Ch2$, or authenticators, $Auth$-$C1$ or $Auth$-$C2$, are not valid. To impersonate the user, AD must obtain access to the user's password. However, to impersonate the server, AD must obtain access to the server's secret S_P. However, these values are kept secret. Consequently, the attacker cannot impersonate the user or the proxy server.

6.6. Replay Attack

A replay attack can be performed if the adversary replays any eavesdropped or intercepted message to forge any legitimate participant. In the A-SIP protocol, the adversary can replay the login request ($M2$) or replay ($M3$) to impersonate the user or the server. However, as explained above, the adversary cannot generate authenticators, $Auth$-$C1$ or $Auth$-$C2$, or the session key, K_{SC}, as shown in Equations (2) and (3). This is because it is not feasible to recover a_1, a_2, b, x, or v from A_1, A_2, A, B_1, B_2, and $Ch1$. Consequently, the adversary fails to authenticate themselves to the client or the server by replaying the login request. Therefore, a replay attack is not applicable for the A-SIP protocol. For the KP-SIP protocol, this type of attack is not possible. This is because the identity of users is protected by encrypting all messages using different session keys.

6.7. Session Teardown Attack

If the attacker discovers credential information from the INVITE message, they can prepare a false BYE message to be sent to the proxy server or one of the user agents to terminate the VoIP session. However, using the S-SIP protocol prevents this attack. User agent C or D uses the session key (K_{CD2}) generated during the authentication process to encrypt the BYE message sent by the caller or callee user agents. Therefore, another party (C or D) can verify the received message. The user agent decrypts the message using the secured shared key K_{CD2}. Then, checks $N4$. If correct, the client processes the BYE message and terminates the session.

6.8. Registration Hijacking Attack

To perform this attack, the adversary must impersonate a valid user agent such as C. The adversary sends an SIP registration message to the proxy server including the URI of C. However, to do so, the adversary must be authenticated at the proxy server through the authentication phase. Then, they encrypt the SIP registration message using the session key and send it to the server. However, as explained in detail in Section 6.1, impersonation of a valid user is not possible. Therefore, the adversary cannot authenticate themselves to the proxy server and cannot obtain the session key. Thus, S-SIP is not vulnerable to registration hijacking attacks.

6.9. Request Spoofing Attack

In this attack, the attacker sends a spoofed INVITE message to fool a legitimate recipient who may believe that they are communicating with another known entity. With the proposed solution, spoofing the S-INVITE message is not possible. Before transmitting the INVITE message, any client must start the process of the mutual authentication phase. If the authentication process fails, the client cannot obtain an authentication ticket or session key. Therefore, because the client's ticket and identity are encrypted using the shared session key, the client cannot construct the S-INVITE message and cannot initiate an SIP call using different identities. Thus, the proposed solution can resist a request spoofing attack.

6.10. Message Tampering Attack

Message tampering attacks occur when an attacker intercepts and modifies packets exchanged between SIP components. This attack is not available in the proposed solution. As shown in Tables 2 and 3, the most sensitive information that the attacker needs to perform this attack is protected by encryption using shared session keys between entities.

6.11. Man-In-The-Middle Attack

The MITM attack is one of the most serious threats to the security and trust of existing VoIP protocols and systems. Using this attack, the attacker can easily wiretap, divert and even hijack VoIP calls by tampering with VoIP signaling and/or media traffic. To execute the MITM attack, the attacker masquerades as a client and an SIP server. However, as explained in Section 6.5, the proposed solution prevents client or server impersonation attacks. Thus, the proposed protocol is not vulnerable to MITM attacks.

6.12. Re-INVITE Attack

After a session is established between user agents, one of them can send a SIP Re-INVITE message to modify the parameters of the session. Therefore, the attacker can perform a DoS attack using the Re-INVITE messages by sending it to one of the user agents. However, as explained above, the S-SIP protocol protects all SIP messages exchanged between user agents and prevents user impersonation attacks. Thus, it prevents Re-INVITE attacks.

7. Formal Security Analysis

ProVerif [53] is a tool used for automatic verification of cryptographic protocols, which is widely used to analyze the security of authentication and key agreement protocols [49,65–68]. In this section, a simulation of the proposed protocols described in Section 5 is performed using ProVerif to illustrate its robustness and correctness under a formal and automated threat model.

Based on the applied π calculus, ProVerif is used to verify key security requirements such as authentication, secrecy, anonymity, and privacy. It can support modeling many cryptographic primitives including one-way functions, encryption and decryption (symmetric and asymmetric), digital signatures, Diffie–Hellman key agreements and many more. Moreover, ProVerif can simulate various attacks. A-SIP and KP-SIP protocols were verified using ProVerif.

For the A-SIP protocol, the complete ProVerif model can be found in [69], and the following explains the main parts of the model in detail, which are shown in Appendix A. We initially defined two channels: a secure channel *SCh* that was used for secure communication between the client and the server, and a public channel *Ch* that was used for public/insecure communication between the client and the server. *SCh* was used in the setup phase and *Ch* was used in the login, authentication, and registration phases. Then, shared session keys, constants, variables, and types used in the proposed A-SIP protocol were defined.

ProVerif defines cryptographic primitives as constructors, destructors and equations. So, the needed constructors (functions) were defined, such as exp, xor, concat, sEnc, and sDEC. Then, the required equations were defined to model the properties of exclusive-OR, modular exponentiation, symmetric encryption, symmetric decryption, extract the first concatenated values, and extract the second concatenated values. Next, four events (UserStarted, UserAuthed, ServerStarted, and ServerEnd) were defined to model the initiation and termination of both client and server processes. The events were used to analyze the security of the proposed protocol.

For A-SIP, we modeled all interactions between the client and the server using two processes: one for the client (*ProcessClient*) and one for the server (*ProcessServer*). The ProVerif script for A-SIP, which included these processes, can be found in [69]. To verify

the model, the two participants could interact by establishing many sessions. Therefore, these two processes were replicated for unbounded parallel executions as follows:

```
(* ======== \Main ======== *)}
process (!ProcessServer |!ProcessClient)
```

Finally, six queries were defined to verify the correctness of the proposed scheme and session key secrecy. These six queries were applied in the main part.

```
(* ===== Queries ===== *)
query attacker(Kc).
query attacker(Ks).
query attacker(Ksc).
query attacker(Kp)
query id : bitstring; inj-event(UserAuthed(id)) ⇒ inj-event(UserStarted(id)).
query id : bitstring; inj-event(ServerEnd(id)) ⇒ inj-event(ServerStarted(id)).
```

ProVerif performed an unbounded number of executions for the model to verify the authenticity and reachability. We executed the modeled processes in ProVerif 2.02. The verification results are as follows:

1. Weak secret PWc is true.
2. Query not attacker(Kc[]) is true.
3. Query not attacker(Ks[]) is true.
4. Query not attacker(Ksc[]) is true.
5. Query not attacker(Kp[]) is true.
6. Query inj-event(UserAuthed(id)) ==> inj-event(UserStarted(id)) is true.
7. Query inj-event(ServerEnd(id)) ==> inj-event(ServerStarted(id)) is true.

When we defined the password as a weak secret, ProVerif tried to prove that the attacker cannot distinguish a correct guess of the secret from an incorrect guess. Therefore, result number 1 shows that the proposed scheme can suppress a password guessing attack. The result numbers 2 to 5 verify that the session keys K_C, K_S, K_{SC}, or K_P were not revealed to the adversary and that secrecy was maintained. The result numbers 6 and 7 show that both *ProcessClient* and *ProcessServer* processes initiated and were completed successfully, respectively, which illustrates the correctness of the proposed authentication protocol.

For the KP-SIP protocol, the complete ProVerif model can be found in [69]. In the ProVerif script, seven queries were defined to verify the correctness of KP-SIP and the session key secrecy. The seven queries were applied in the main part and are as follows:

```
(* ======== Queries ======== *)
query attacker(Ksc).
query attacker(Ksd).
query attacker(Kcd1).
query attacker(Kcd2).
query URIc : bitstring; inj-event(C_End(URIc)) ⇒ inj-event(C_End(URIc)).
query URId : bitstring; inj-event(C_End(URId)) ⇒ inj-event(C_End(URId)).
query id : bitstring; inj-event(ServerEnd(id)) ⇒ inj-event(ServerStarted(id)).
```

After performing an unbounded number execution for the model to verify the authenticity and reachability, ProVerif showed the following verification results for the KP-SIP model:

1. Query not attacker(Ksc[]) is true.
2. Query not attacker(Ksd[]) is true.
3. Query not attacker(Kcd1[]) is true.
4. Query not attacker(Kcd2[]) is true.

5. Query inj-event(C_End(URIc_1)) ==> inj-event(C_End(URIc_1)) is true.
6. Query inj-event(C_End(URId_2)) ==> inj-event(C_End(URId_2)) is true.
7. Query inj-event(ServerEnd(id)) ==> inj-event(ServerStarted(id)) is true.

The result numbers 1 to 4 verify that the session keys K_{SC}, K_{CD1}, K_{SD}, or K_{CD2} were not revealed to the adversary and that secrecy was maintained. The result numbers 5 to 7 show that all processes for *C*, *D* and *S* initiated and were completed successfully, which illustrates the correctness of the proposed protocol.

8. Performance Analysis

8.1. Performance Comparison

In this section, we compare the security features and performance of the proposed scheme with other related schemes [39,41–43,45,48–51]. First, we compared the proposed authentication protocol A-SIP with related protocols considering various security features. Table 4 demonstrates the analysis of the security features for A-SIP in comparison with the related works. The proposed protocol was secure against all mentioned attacks, and can provide security requirements, including perfect forward secrecy and stolen-verifier attacks. In other words, the proposed scheme provides a high level of security compared to the related authentication protocols.

Table 4. Comparison of security features.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12
[39]	N	Y	Y	Y	N	N	Y	Y	Y	Y	Y	N
[45]	Y	Y	Y	N	N	Y	N	Y	Y	Y	Y	Y
[41]	Y	Y	Y	N	N	Y	N	Y	N	Y	Y	Y
[42]	Y	Y	Y	Y	N	N	Y	Y	N	Y	Y	Y
[43]	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y	Y
[48]	Y	Y	Y	N	N	N	Y	Y	Y	Y	Y	Y
[49]	N	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y
[50]	N	Y	Y	Y	Y	N	Y	Y	Y	N	Y	Y
[51]	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y
SRP	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
A-SIP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

F1: offline password guessing attack resistance; F2: stolen verifier attack resistance; F3: Denning–Sacco attack resistance; F4: replay attack resistance; F5: user impersonation attack resistance; F6: server impersonation attack resistance; F7: privileged insider attack resistance; F8: MITM attack resistance; F9: providing mutual authentication; F10: providing perfect forward secrecy; F11: providing known key secrecy; F12: session-key temporary information attack resistance. 'Y': scheme provides the feature; 'N': scheme does not provide the feature.

For performance comparison, the computational cost for each related authentication protocol was calculated using the primitive arithmetic and cryptographic operation timings. Table 5 shows the notation used for different cryptographic operations and the arithmetic mean and standard deviation of the computation cost of each operation. These cryptographic operations were implemented using Python programming language using the PyCryptodome 3.11.0 library on a personal computer with 8 GB RAM, an Intel Core i5-10210U CPU @ 2.1 GHz, and a 64-bit Windows 10 Professional operating system. Each operation was executed thousands of times until we obtained a 95% confidence interval with 2% error.

To compute the computational cost of symmetric encryption and decryption, we used AES with a 256-bit key and a message with a size of 512 bytes. For ECC point multiplication and addition, the standard elliptic curve Secp256k1 was used, where the modulus p and the order n were 256-bit numbers. This curve provides 128-bit security strength and is one of the curves adopted by many applications, such as OpenSSH and Bitcoin.

In authentication schemes, the login and authentication phases are executed more frequently than other phases that are performed only once. Therefore, the computational costs of only the login and authentication phases were considered in the performance comparison. A detailed performance evaluation of each related scheme is shown in Table 6.

This table shows the arithmetic and cryptographic operations performed on both the client side and the server side and the estimated computational cost for each protocol. The timings in Table 6 were calculated using the primitive arithmetic and cryptographic operation timings given in Table 5. The computational cost of lightweight operations, such XOR and concatenation, were ignored.

Table 5. Mean and standard deviation for execution time for different cryptographic operations.

Symbol	Operation	Mean (µs)	Stan. Dev. (µs)
T_H	One-way hash function (SHA-1)	16.5	4.8
T_{PM}	Elliptic curve point multiplication	32,646.2	365.5
T_{PA}	Elliptic curve point addition	838.7	26.5
T_{Exp}	Modular exponentiation	345.4	34.8
T_{SED}	Symmetric Key encryption/decryption	157.5	12.8
T_R	Random number generation	17.4	1.6

Table 6. Comparison of the computational cost.

Schemes	Side of Operations	Operations	Cost (ms)	Total (ms)
[39]	Client	$3T_H + T_R + 3T_{PM} + T_{PA}$+$T_{SED}$	99.001	197.620
	Server	$2T_H + 2T_R + 3T_{PM} + T_{SED}$	98.619	
[45]	Client	$6T_H + T_R + 4T_{PM} + T_{PA}$	131.539	263.868
	Server	$3T_H + 2T_R + 4T_{PM} + 2T_{PA}$	132.329	
[41]	Client	$4T_H + T_R + 3T_{PM}$	98.022	196.044
	Server	$4T_H + T_R + 3T_{PM}$	98.022	
[42]	Client	$4T_H + T_R + 2T_{PM}$	65.375	130.767
	Server	$5T_H + T_R + 2T_{PM}$	65.392	
[43]	Client	$4T_H + T_R + 3T_{PM}$	98.022	163.381
	Server	$3T_H + T_R + 2T_{PM}$	65.359	
[48]	Client	$4T_H + T_R + 3T_{PM} + T_{PA}$	98.860	196.882
	Server	$4T_H + 2T_R + 3T_{PM}$	98.022	
[49]	Client	$5T_H + T_R + 3T_{PM} + T_{PA}$	98.877	197.265
	Server	$5T_H + 3T_R + 3T_{PM} + 2T_{SED}$	98.388	
[50]	Client	$5T_H + T_R + 2T_{SED}$	0.572	1.443
	Server	$3T_H + 2T_R + 5T_{SED}$	0.871	
[51]	Client	$7T_H + 2T_R + 2T_{SED}$	0.465	1.527
	Server	$5T_H + 2T_R + 6T_{SED}$	1.062	
SRP	Client	$5T_H + T_R + 3T_{Exp}$	13.136	26.239
	Server	$3T_H + T_R + 3T_{Exp}$	13.103	
A-SIP	Client	$8T_H + 2T_R + 6T_{Exp} + T_{SED}$	26.363	44.002
	Server	$7T_H + T_R + 4T_{Exp} + T_{SED}$	17.639	

As shown in Table 6, the proposed scheme was fourth in terms of computational cost. The authentication schemes introduced in [50,51] had the lowest costs. However, as explained in Section 2, these schemes do not provide the perfect forward secrecy security requirement.

8.2. S-SIP Overhead

To characterize the performance of the S-SIP protocol and to indicate its overhead compared to SIP and related protocols based on TLS, we implemented an experimental testbed shown in Figure 6 based on the scenario depicted in Figure 1A. The testbed was a wide area network consisting of three 100 Mbps LANs. The user agents were connected to the first and second LANs, whereas the proxy/authentication server resided on the third LAN. As shown in Figure 6, the user agents and the server were connected through the internet over 500 Mbps connections.

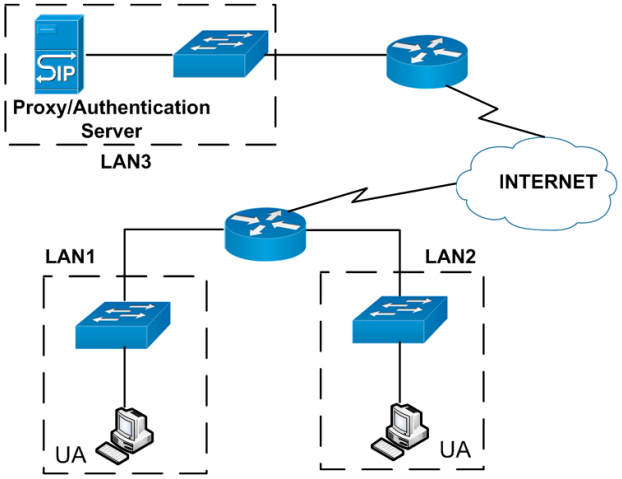

Figure 6. Testbed network architecture.

The two user agents had the same specifications explained in Section 8.1. The server was a PC that had an Intel core i7 2.4 GHz processor and 16 GB RAM with a Windows 10 Pro 64-bit operating system. The S-SIP protocol was implemented using Python. The user agents and the server were implemented based on the scenario shown in Figure 1, where exchanging RTP messages were not considered because it was out of the scope of this work. In all experiments, for symmetric encryption and decryption, we used AES with a 256-bit key.

To compare between S-SIP and TLS-based techniques used to secure SIP, we suppose that SIPS/TLS is used for securing SIP messages and SRP is used as a password-based authentication protocol. We called this method as SRP–TLS. SRP–TLS was implemented using Python, where TLS version 1.2 with cipher suit ECDHE-RSA-AES256-GCM-SHA384 were used.

Because S-SIP and SRP–TLS use more messages than SIP, their overhead depends on the message round trip time (RTT) between clients and the server. Therefore, we considered two scenarios: small and large RTT. Thus, the server was placed in an area geographically close or far from the user agents to represent these two scenarios.

The overheads of S-SIP and SRP-TLS were measured using two performance metrics: the authentication and registration time T_A and the call setup and teardown time T_S. T_A is the time required to authenticate and register a user agent to the server. T_S is the time required to setup and tear down the call between the caller and callee.

For all measured performance metrics, experiments were repeated until we obtained 95% confidence intervals with 2% error. For S-SIP, SRP–TLS and SIP, the mean and standard deviation were measured for each performance metric. To measure the actual overhead, we suppose that SIP does not use any authentication mechanism. For small and large RTT scenarios, the measured ping times between the user agents and the server were 32 and 227 ms, respectively. The results are shown in Table 7.

Table 7. T_A and T_S (ms) for protocols SIP, SRP-TLS and S-SIP.

		RTT = 32 ms			RTT = 227 ms		
		SIP	**S-SIP**	**SRP-TLS**	**SIP**	**S-SIP**	**SRP-TLS**
T_A (ms)	Mean	33.9	146.6	275.4	229.4	733.1	1389.1
	Stan. Dev.	0.3	2.7	7.3	3.4	9.5	11.4
T_S (ms)	Mean	152.3	171.9	166.9	1032.1	1149.5	1102.8
	Stan. Dev.	2.5	3.6	6.8	12.4	14.8	14.2
Overhead (ms)		NA	132.3	256.1	NA	621.1	1230.4

Compared to T_S, the overhead in T_A was larger mainly due to the authentication process and handshake mechanisms supported by S-SIP and SRP–TLS. For small RTT, the total overheads of S-SIP and SRP-TLS were 132.3 and 256.1 ms, respectively, whereas for large RTT, they were 621.1 and 1230.4 ms, respectively. In the case of small and large RTTs, the total overhead of SRP–TLS was nearly double that of S-SIP. The main overhead in the SIPS protocol was due to the TLS handshake phase and verification of digital certificates. In addition, SIPS protocol constructs TLS secure tunnels among each hop in the path from the client to the final recipient. Therefore, increasing the number of proxy servers between user agents greatly increased the overhead of SIPS.

9. Conclusions

This work introduced a new protocol for securing SIP called S-SIP, which can thwart most SIP attacks. S-SIP consists of two protocols: A-SIP and KP-SIP. The A-SIP protocol is an authentication protocol that provides mutual authentication for SIP entities. The KP-SIP protocol is used to secure SIP signaling messages and to exchange session keys and authentication tickets between SIP entities. A-SIP is based on the SRP protocol, which is one of the standard password-based authentication protocols supported by TLS. We informally analyzed the security issues of SRP. We showed that it is vulnerable to offline password guessing, stolen-verifier, and Denning–Sacco attacks. Therefore, we proposed the A-SIP protocol to overcome these security flaws in SRP. In addition, through informal security analysis, we showed that S-SIP can thwart many SIP-based attacks, such as session teardowns, registration hijacking, request spoofing, message tampering, and re-invite attacks. Additionally, we verified the security of S-SIP through formal analysis using the ProVerif tool. Moreover, we compared A-SIP with multiple related authentication schemes in terms of security and performance. Comparisons showed that A-SIP has more security features and lower computational costs. SIPS is a standard protocol for securing SIP based on TLS. To characterize the overhead of S-SIP compared to SIP and SRP–TLS, its performance was analyzed. Compared to SIP, in the case of small and large RTTs, the overheads of S-SIP were 132.3 and 621.1 ms, respectively. For SRP–TLS, for small and large RTTs, the overheads were 256.1 and 1230.4 ms, respectively. The results showed that the performance of S-SIP was better than SRP-TLS. In addition, increasing the number of proxy servers between user agents greatly increased the overhead of SRP–TLS. As a future work, KP-SIP can be extended to secure connections among SIP servers owned by the same or different telecoms/providers.

Author Contributions: Methodology, O.Y.; Validation, U.A.; Formal and informal analysis, O.Y.; Investigation, U.A.; Funding acquisition, U.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

ProVerif defines cryptographic primitives as constructors, destructors and equations. Table A1 shows the constructors that we defined to model the proposed protocol. The following shows the main parts of the ProVerif code that are explained in Section 7.

Table A1. Constructors defined in the ProVerif model of A-SIP.

Constructor	Function
h	One-way hash function
mult	Multiplication of two integers
mult3	Multiplication of three integers
div	Integer division
concat	Concatenation 2, 3, 4 or 6 values
exp	Exponential operation
xor	Exclusive-OR
sEnc	Symmetric encryption
sDec	Symmetric decryption
gKey	Converting from the bitstring type to the key type
BitKey	Converting from the key type to the bitstring type
getfirst	Extract the first of concatenated values
getsec	Extract the second of concatenated values

```
(* === channels ==== *)
free SCh : channel [private].
free Ch : channel.
(* Shared keys *)
free Kc : key [private].
free Ks : key [private].
free Ksc : key [private].
free Kp : key [private].
(* constants & Variables *)
const g : bitstring.
const p : bitstring.
const IDc : bitstring.
free IPs : bitstring.
free Sp : bitstring [private].
free Pwc : bitstring [private].
weaksecret PWc.
(* ==== Types ==== *)
type key.
(* ==== Functions ==== *)
fun h(bitstring) : bitstring.
fun mult(bitstring, bitstring) : bitstring.
fun mult3(bitstring, bitstring, bitstring) : bitstring.
fun div(bitstring, bitstring) : bitstring.
fun concat(bitstring, bitstring) : bitstring.
fun concat3(bitstring, bitstring, bitstring) : bitstring.
fun concat4(bitstring, bitstring, bitstring, bitstring) : bitstring.
fun concat6(bitstring, bitstring, bitstring, bitstring, bitstring, bitstring) :
    bitstring.
fun exp(bitstring, bitstring) : bitstring.
fun xor(bitstring, bitstring) : bitstring.
fun sEnc (key, bitstring) : bitstring.
fun sDec (key, bitstring) : bitstring.
fun gKey(bitstring) : key.
fun BitKey(key) : bitstring.
```

fun getfirst(bitstring) : bitstring.
fun getsecond(bitstring) : bitstring.
(* ======== ***Equations*** ======== *)
equation forall x : bitstring, y : bitstring; exp(exp(g, x), y) = exp(exp(g, y), x).
equation forall m : bitstring, k : key; sDec (k, sEnc (k, m)) = m.
equation forall m : bitstring, k : key; sEnc (k, sDec (k, m)) = m.
equation forall m : bitstring, k : bitstring; xor (k, xor(k, m)) = m.
quation forall m : bitstring, n : bitstring; getfirst (concat(m, n)) = m.
equation forall m : bitstring, n : bitstring; getsec (concat(m, n)) = n.
(* ==== ***Events*** ==== *)
event UserStarted(bitstring).
event UserAuthed(bitstring).
event ServerStarted(bitstring).
event ServerEnd(bitstring).
(* ===== ***Main*** ===== *)
process (!*ProcessServer* |!*ProcessClient*)
(* ===== ***Queries*** ===== *)
query *attacker*(Kc).
query *attacker*(Ks).
query *attacker*(Ksc).
query *attacker*(Kp)
query id : bitstring; **inj-event**(UserAuthed(id)) ⇒ **inj-event**(UserStarted(id)).
query id : bitstring; **inj-event**(ServerEnd(id)) ⇒ **inj-event**(ServerStarted(id)).

References

1. Schulzrinne, H.; Casner, S.; Frederick, R.; Jacobson, V. *RFC 3550-RTP: A Transport Protocol for Real-Time Applications*; IETF: Fremont, CA, USA, 2003.
2. Rosenberg, J.; Schulzrinne, H.; Camarillo, G.; Johnston, A.; Peterson, J.; Sparks, R.; Handley, M.; Schooler, E. *RFC 3261-Sip: Session Initiation Protocol*; IETF: Fremont, CA, USA, 2002.
3. Tam, K.; Goh, H. Session initiation protocol. In Proceedings of the 2002 IEEE International Conference on Industrial Technology, 2002, IEEE ICIT '02, Bangkok, Thailand, 11–14 December 2002; pp. 1310–1314.
4. Chiang, W.K.; Chang, W.Y. Mobile-initiated network-executed SIP-based handover in IMS over heterogeneous accesses. *Int. J. Commun. Syst.* **2010**, *23*, 1268–1288. [CrossRef]
5. Cho, K.; Pack, S.; Kwon, T.T.; Choi, Y. An extensible and ubiquitous RFID management framework over next-generation network. *Int. J. Commun. Syst.* **2010**, *23*, 1093–1110. [CrossRef]
6. Keromytis, A.D. A Look at VoIP Vulnerabilities. *Usenix Secur. Artic.* **2010**, *35*, 41–50.
7. Keromytis, A.D. A Comprehensive Survey of Voice over IP Security Research. *IEEE Commun. Surv. Tutor.* **2012**, *14*, 514–537. [CrossRef]
8. Ahson, A.S.; Ilyas, M. *Sip Handbook Services, Technologies, And Security of Session Initiation Protocol*; CRC Press: Boca Raton, FL, USA, 2009.
9. Ahson, A.S.; Ilyas, M. *VoIP Handbook, Applications, Technologies, Reliability, and Security*; CRC Press: Boca Raton, FL, USA, 2009.
10. Sisalem, D.; Floroiu, J.; Kuthan, J.; Abend, U.; Schulzrinne, H. *SIP Security*; John Wiley & Sons Ltd.: Hoboken, NJ, USA, 2009.
11. Franks, J.; Hallam-Baker, P.; Hostetler, J.; Lawrence, S.; Leach, P.; Luotonen, A.; Stewart, L. *RFC 2617-HTTP Authentication: Basic and Digest Access Authentication*; IETF: Fremont, CA, USA, 1999.
12. Kent, S.; Seo, K. *RFC 4301-Security Architecture for the Internet Protocol*; IETF: Fremont, CA, USA, 2005.
13. Dierks, T.; Rescorla, E. *RFC 5246-The Transport Layer Security (TLS) Protocol*; IETF: Fremont, CA, USA, 2008.
14. Ramsdell, B. *RFC 3851-Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification*; IETF: Fremont, CA, USA, 2004.
15. Nguyen, K.T.; Laurent-Maknavicius, M.; Oualha, N.J.A.H.N. Survey on secure communication protocols for the Internet of Things. *Ad Hoc Netw.* **2015**, *32*, 17–31. [CrossRef]
16. Haase, B.; Labrique, B. Aucpace: Efficient verifier-based PAKE protocol tailored for the IIOT. *IACR Cryptol. Eprint Arch.* **2018**, *2018*, 286. [CrossRef]
17. Sebek, F.; Petri, O.; Sebek, F. *A Comparison of the Password-Authenticated Key Exchange Protocols, SRP-6a and PAKE2+*; Technical Report; Kth Royal Institute of Technology, School of Electrical Engineering and Computer Science: Stockholm, Sweden, 2019.
18. Shin, S.; Kobara, K. Security Analysis of Password-Authenticated Key Retrieval. *IEEE Trans. Dependable Secur. Comput.* **2017**, *14*, 573–576.

19. Jarecki, S.; Krawczyk, H.; Xu, J. OPAQUE: An Asymmetric PAKE Protocol Secure Against Pre-computation Attacks. In *Advances in Cryptology—EUROCRYPT 2018, Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, 29 April–3 May 2018*; Springer International Publishing: Cham, Switzerland, 2018.
20. Bellovin, S.M.; Merritt, M. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In Proceedings of the CCS93: 1st ACM Conference on Communications and Computing Security, Fairfax, VA, USA, 3–5 November 1993; pp. 244–250.
21. Boyd, C.; Mathuria, A.; Stebila, D. *Protocols for Authentication and Key Establishment*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2003.
22. Hao, F.; Ryan, P.Y.A. Password Authenticated Key Exchange by Juggling. In *Security Protocols XVI, Proceedings of the 16th International Workshop on Security Protocols, Cambridge, UK, 16–18 April 2008*; Springer: Berlin/Heidelberg, Germany, 2008.
23. Yoneyama, K. Cross-Realm Password-Based Server Aided Key Exchange. In *Information Security Applications*; Springer: Berlin/Heidelberg, Germany, 2011.
24. Wu, T.D. The secure remote password protocol. *NDSS* **1998**, *98*, 97–111.
25. Wu, T. *RFC 2945-The SRP Authentication and Key Exchange System*; IETF: Fremont, CA, USA, 2000.
26. Taylor, T.W.D.; Mavrogiannopoulos, N.; Perrin, T. *RFC 5054-Using the Secure Remote Password (SRP) Protocol for TLS Authentication*; IETF: Fremont, CA, USA, 2007.
27. *IEEE Std 1363.2™-2008*; IEEE Standard Specification for Password-Based Public-Key Cryptographic Techniques. IEEE Computer Society: Piscataway, NJ, USA, 2008; 140p.
28. Tom, W. Official Website for SRP. Available online: http://srp.stanford.edu/ (accessed on 7 March 2022).
29. Yang, C.-C.; Wang, R.-C.; Liu, W.-T. Secure authentication scheme for session initiation protocol. *Comput. Secur.* **2005**, *24*, 381–386. [CrossRef]
30. Huang, H.-F. A new efficient authentication scheme for Session Initiation Protocol. In Proceedings of the 9th Joint International Conference on Information Sciences (JCIS-06), Kaohsiung, Taiwan, 8–11 October 2006; Atlantis Press: Amsterdam, The Netherlands, 2006; pp. 402–404.
31. Jo, H.; Lee, Y.; Kim, M.; Kim, S.; Won, D. Off-Line Password-Guessing Attack to Yang's and Huang's Authentication Schemes for Session Initiation Protocol. In Proceedings of the 2009 Fifth International Joint Conference on INC, IMS and IDC, Seoul, Republic of Korea, 25–27 August 2009; pp. 618–621.
32. Durlanik, A.; Sogukpinar, I. SIP authentication scheme using ECDH. *Proc. World Acad. Sci. Eng. Technol.* **2005**, *8*, 350–353.
33. Wu, L.; Zhang, Y.; Wang, F. A new provably secure authentication and key agreement protocol for SIP using ECC. *Comput. Stand. Interfaces* **2009**, *31*, 286–291. [CrossRef]
34. Koblitz, N. Elliptic curve cryptosystems. *Math. Comput.* **1987**, *48*, 203–209. [CrossRef]
35. Yoon, E.-J.; Yoo, K.-Y.; Kim, C.; Hong, Y.-S.; Jo, M.; Chen, H.-H. A secure and efficient SIP authentication scheme for converged VoIP networks. *J. Comput. Commun.* **2010**, *33*, 1674–1681. [CrossRef]
36. Pu, Q. Weaknesses of SIP Authentication Scheme for Converged VoIP Networks. Cryptology ePrint Archive. Paper 2010/464. Available online: https://eprint.iacr.org/2010/464 (accessed on 7 March 2022).
37. Tsai, J.L. Efficient nonce-based authentication scheme for session initiation protocol. *Int. J. Netw. Secur.* **2009**, *8*, 312–316.
38. Yoon, E.-J.; Shin, Y.-N.; Jeon, I.-S.; Yoo, K.-Y. Robust mutual authentication with a key agreement scheme for the session initiation protocol. *IETE Tech. Rev.* **2010**, *27*, 203–213. [CrossRef]
39. Xie, Q. A new authenticated key agreement for session initiation protocol. *Int. J. Commun. Syst.* **2012**, *25*, 47–54. [CrossRef]
40. Farash, M.S.; Attari, M.A. An enhanced authenticated key agreement for session initiation protocol. *Inf. Technol. Control* **2013**, *42*, 333–342. [CrossRef]
41. Zhang, Z.; Qi, Q.; Kumar, N.; Chilamkurti, N.; Jeong, H.-Y. A secure authentication scheme with anonymity for session initiation protocol using elliptic curve cryptography. *Multimed. Tools Appl.* **2015**, *74*, 3477–3488. [CrossRef]
42. Lu, Y.; Li, L.; Peng, H.; Yang, Y. A secure and efficient mutual authentication scheme for session initiation protocol. *Peer-Peer Netw. Appl.* **2016**, *9*, 449–459. [CrossRef]
43. Chaudhry, S.; Khan, I.; Irshad, A.; Ashraf, M.U.; Khan, M.K.; Ahmad, H.F. A provably secure anonymous authentication scheme for Session Initiation Protocol. *Secur. Commun. Netw.* **2016**, *9*, 5016–5027. [CrossRef]
44. Kumari, S.; Karuppiah, M.; Das, A.K.; Li, X.; Wu, F.; Gupta, V. Design of a secure anonymity-preserving authentication scheme for session initiation protocol using elliptic curve cryptography. *J. Ambient. Intell. Humaniz. Comput.* **2018**, *9*, 643–653. [CrossRef]
45. Zhang, L.; Tang, S.; Cai, Z. Efficient and flexible password authenticated key agreement for Voice over Internet Protocol Session Initiation Protocol using smart card. *Int. J. Commun. Syst.* **2013**, *27*, 2691–2702. [CrossRef]
46. Irshad, A.; Sher, M.; Rehman, E.; Ch, S.A.; Hassan, M.U.; Ghani, A. A single round-trip SIP authentication scheme for Voice over Internet Protocol using smart card. *Multimed. Tools Appl.* **2015**, *74*, 3967–3984. [CrossRef]
47. Arshad, H.; Nikooghadam, M. Security analysis and improvement of two authentication and key agreement schemes for session initiation protocol. *J. Supercomput.* **2015**, *71*, 3163–3180. [CrossRef]
48. Tu, H.; Kumar, N.; Chilamkurti, N.; Rho, S. An improved authentication protocol for session initiation protocol using smart card. *Peer-Peer Netw. Appl.* **2015**, *8*, 903–910. [CrossRef]
49. Chaudhry, S.A.; Naqvi, H.; Sher, M.; Farash, M.S.; Hassan, M.U. An improved and provably secure privacy preserving authentication protocol for SIP. *Peer-Peer Netw. Appl.* **2017**, *10*, 1–15. [CrossRef]

50. Nikooghadam, M.; Jahantigh, R.; Arshad, H. A lightweight authentication and key agreement protocol preserving user anonymity. *Multimed. Tools Appl.* **2017**, *76*, 13401–13423. [CrossRef]
51. Ravanbakhsh, N.; Mohammadi, M.; Nikooghadam, M. Perfect forward secrecy in VoIP networks through design a lightweight and secure authenticated communication scheme. *Multimed. Tools Appl.* **2019**, *78*, 11129–11153. [CrossRef]
52. Nikooghadam, M.; Amintoosi, H. Perfect forward secrecy via an ECC-based authentication scheme for SIP in VoIP. *J. Supercomput.* **2020**, *76*, 3086–3104. [CrossRef]
53. Abadi, M.; Blanchet, B.; Comon-Lundh, H. Models and Proofs of Protocol Security: A Progress Report. In *Computer Aided Verification, Proceedings of the 21st International Conference on Computer Aided Verification, Grenoble, France, 26 June–2 July 2009*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2009; pp. 35–49.
54. Audet, F. *The Use of the SIPS URI Scheme in the Session Initiation Protocol (SIP)*; RFC 5630; IETF: Fremont, CA, USA, 2009.
55. Abubakar, M.; Jaroucheh, Z.; Al Dubai, A.; Buchanan, B. Blockchain-Based Authentication and Registration Mechanism for SIP-Based VoIP Systems. In Proceedings of the 5th Cyber Security in Networking Conference (CSNet), Abu Dhabi, United Arab Emirates, 12–14 October 2021; pp. 63–70.
56. Aldahwan, N.; Alghazzawi, D. Use of Blockchain in Public Key Infrastructure (PKI): A Systematic Literature Review. *Int. J. Comput. Sci. Inf. Secur.* **2020**, *18*, 106–111.
57. Johnston, A. *SIP: Understanding the Session Initiation Protocol*, 3rd ed.; Artech House: Norwood, MA, USA, 2009.
58. Kulkarni, L. VoIP Security: A Performance and Cost-benefit Analysis. *Inf. Technol. Ind.* **2021**, *8*, 34–42. [CrossRef]
59. Bates, R.J. *Securing VoIP: Keeping Your VoIP Network Safe*; Elsevier Inc.: Amsterdam, The Netherlands, 2015.
60. Omar, M.I.; Kiprut, C.W.; Kimwele, M.W. Securing the IP Multimedia Subsystem with IPsec and HTTP Digest. *Int. J. Comput.* **2017**, *26*, 117–128.
61. Farley, R.; Wang, X. VoIP Shield: A transparent protection of deployed VoIP systems from SIP-based exploits. In Proceedings of the 2012 IEEE Network Operations and Management Symposium, Maui, HI, USA, 16–20 April 2012; pp. 486–489.
62. Basem, B.; Ghalwash, A.Z.; Sadek, R.A. Multilayer Secured SIP Based VoIP Architecture. *Int. J. Comput. Theory Eng.* **2015**, *7*, 453–462. [CrossRef]
63. Sherman, A.T.; Lanus, E.; Liskov, M.; Zieglar, E.; Chang, R.; Golaszewski, E.; Wnuk-Fink, R.; Bonyadi, C.J.; Yaksetig, M.; Blumenfeld, I. Formal Methods Analysis of the Secure Remote Password Protocol. In *Logic, Language, and Security*; Springer: Cham, Switzerland, 2020.
64. Arshad, H.; Nikooghadam, M. An efficient and secure authentication and key agreement scheme for session initiation protocol using ECC. *Multimed. Tools Appl.* **2016**, *75*, 181–197. [CrossRef]
65. Chen, C.-M.; Xiang, B.; Wu, T.-Y.; Wang, K.-H. An Anonymous Mutual Authenticated Key Agreement Scheme for Wearable Sensors in Wireless Body Area Networks. *Appl. Sci.* **2018**, *8*, 1074. [CrossRef]
66. Wu, F.; Xu, L.; Kumari, S.; Li, X.; Shen, J.; Choo, K.-K.R.; Wazid, M.; Das, A.K. An efficient authentication and key agreement scheme for multi-gateway wireless sensor networks in IoT deployment. *J. Netw. Comput. Appl.* **2017**, *89*, 72–85. [CrossRef]
67. Abbasinezhad-Mood, D.; Nikooghadam, M. Efficient Anonymous Password-Authenticated Key Exchange Protocol to Read Isolated Smart Meters by Utilization of Extended Chebyshev Chaotic Maps. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4815–4828. [CrossRef]
68. Abbasinezhad-Mood, D.; Nikooghadam, M. Design and hardware implementation of a security-enhanced elliptic curve cryptography based lightweight authentication scheme for smart grid communications. *Future Gener. Comput. Syst.* **2018**, *84*, 47–57. [CrossRef]
69. Younes, O. ProVerif Model for S-SIP Protocol. Available online: https://drive.google.com/drive/folders/1Bks5GwfWbt3v1qgqKFH0mREgzhf3J3Bj?usp=sharing (accessed on 20 March 2022).