Journal of Universal Computer Science, vol. 23, no. 10 (2017), 969-991
submitted: 21/3/17, accepted: 20/10/17, appeared: 28/10/17

OnionSIP: Preserving Privacy in SIP with Onion Routing

Alexandros Fakis
(Dept. of Information and Communication Systems
Engineering, University of the Aegean, Samos, Greece
alfa@aegean.gr)

Georgios Karopoulos
(Dept. of Informatics and Telecommunications
National and Kapodistrian University of Athens, Athens, Greece
gkarop@di.uoa.gr)

Georgios Kambourakis
(Dept. of Information and Communication Systems
Engineering, University of the Aegean, Samos, Greece
Computer Science Department,
George Mason University, Fairfax, VA 22030, USA
gkamb@aegean.gr)

Abstract: While more and more users turn to IP-based communication technologies, privacy and anonymity remain largely open issues. One of the most prominent VoIP protocols for multimedia session management is SIP which, despite its popularity, suffers from security and privacy flaws. As SIP messages are sent in plain text, user data are exposed to intermediate proxies and eavesdroppers. As a result, information about users participating in a call can leak from header data, which cannot be omitted since they are needed for the correct routing of SIP messages to their final destination. Even more, traffic analysis attacks can be mounted with data stemming from lower layers. To redress this kind of problems, privacy can be achieved either by the construction of a lower level tunnel (via the use of SSL or IPsec protocols) or by employing a custom-tailored solution. However, SSL and IPsec are known for leading to undesirable, non affordable delays, and thus the need for a SIP-oriented solution is preferable. In the context of this article, we evaluate three alternative solutions to encounter the above issues. More specifically, we use two well-known anonymity networks, Tor and I2P, for secluding both caller's and callee's actions by securing SIP messages content. As a third solution, we present our proposal for preserving privacy in SIP signaling, by using an onion-routing approach, where selected sensitive fields of SIP messages are encrypted using either asymmetric or symmetric encryption. We compare these three alternatives in terms of performance, mentioning the pros and cons that come up with each proposal. Our work also presents the reasons why other existing anonymity networks fail to be considered as appropriate for preserving anonymity in SIP.

Key Words: Session Initiation Protocol, Privacy, Anonymity, Tor, I2P
Category: H.4.3, H.5.1

1 Introduction

The need for ways of communication with lower cost and less maintenance than traditional ones based on Public Switched Telephone Network (PSTN), led most users and organizations to turn to more flexible solutions, like Voice-over-IP (VoIP). The most concrete VoIP benefit that gets easily noticed against PSTN is the cost savings, although the high scalability and many free added-value features cannot be ignored either. One of the most prominent VoIP protocols offering multimedia services is the Session Initiation Protocol (SIP) [Rosenberg et al., 2010], which is an application layer, text-based protocol, used for signaling and managing multimedia sessions. Although SIP is a popular protocol, it still suffers from basic privacy weaknesses and security issues, e.g., information about the users participating in a voice call can be easily exposed to a third party user.

Due to its text-based nature, SIP suffers from two main weaknesses, which are the disclosure of: (a) user identities, and (b) user IP addresses. One of the main reasons why SIP is not considered a secure protocol is because the contents of each SIP message are transferred in plain text, rather than being encrypted in any way. Having that in mind, any malicious user or intermediate proxy, which is able to read the SIP message contents, can reveal sensitive information about the users participating in the call. More specifically, a malicious user may reveal: (a) the caller's name, username, and IP address, and (b) the callee's name, username, and IP address. These information derive from the unprotected `From, To, Contact` and `Call-ID` fields. An example of a typical SIP message is shown in Table 1; here some data, like branch and tag values, were omitted for readability.

```
INVITE sip:alfakis@agn.org SIP/2.0
Via: SIP/2.0/UDP pc8.agn.org;branch=...
Max-Forwards: 70
To: Alex <sip:alfakis@agn.org>
From: George <sip:geokarop@agn.org>;tag=...
Call-ID: a84b4c76e66710@pc8.agn.org
CSeq: 314159 INVITE
Contact: <sip:geokarop@194.252.165.10>
Content-Type: application/sdp
Content-Length: 142
```

Table 1: SIP message header format

Apart from the fact that SIP messages are transferred in plain text format and user IPs are exposed through SIP header fields, IPs are also disclosed from packet exchange on the network layer. In that case, any observer has the ability to obtain the IP addresses of both ends just by tracking the headers of the IP packets conveying SIP messages. Thus, it is meaningless to try and hide any IP address appearing in SIP header fields, for instance the ones in `From` and `To` fields. Obfuscating only the application layer information would have no actual effect, as the observer can easily correlate a message leaving the caller with another message that is reaching the callee.

Although standard security protocols, like IPsec and SSL, constitute an effective solution for initializing a VoIP session securely, they come with significant implementation demands, while call setup delays are not negligible either. In [Salama et al., 2009], the delay of a SIP call setup is measured in two scenarios: (a) using plain IP, and (b) using IPsec. In SIP over plain IP, delays are between 0.5 and 1 second, while the utilization of IPsec for initiating a call can approach a delay of 7.5 seconds, using the encryption scheme only. The introduced delays concern the SIP protocol and occur only once during session establishment, without affecting the actual call data which are exchanged in a later phase.

The lack of a truly efficient and easily deployable privacy solution for SIP motivated us to search for a different approach to deal with the user identity and IP address disclosure issues. Our interest was spurred by how a VoIP protocol, like SIP, would act if an anonymizing network intervened. Our research started from the two most widespread anonymizing networks, Tor [Dingledine et al., 2004] and I2P [Zantout and Haraty, 2011]. At first, Tor offers the ability to a user to use it as a SOCKS proxy, making it a suitable solution for anonymizing SIP traffic. On the other hand, as I2P works slightly different from Tor, one will need an outbound proxy to communicate with non-I2P users. At the moment, I2P lacks SOCKS proxy support, so the only alternative way to anonymize traffic is by using its HTTP proxies.

Even though we examined other solutions, apart from Tor and I2P, they cannot support SIP anonymization with reasonable deployment effort. Some are in development or in early experimental stage, like Tarzan [Freedman and Morris, 2002] or Riffle [Kwon, 2015], while others do not provide an applicable proxy to pass SIP traffic through. The above facts restrict the diverse techniques to anonymize SIP traffic.

In search of a simpler and more efficient solution, we decided to design and implement an encryption-based privacy-preserving scheme, where all SIP sensitive fields are encrypted. The same information can later be recovered only by authorized parties, i.e., the proxies of each user, as well as the users participating in the call. In the proposed onion-routing based scheme, sensitive information contained in a SIP message is encrypted in a multilayer fashion, capitalising on the onion routing concept [Syverson et al., 2001]. This way, inbound and outbound SIP messages cannot be correlated, as long as the application layer fields are hidden. However, traffic analysis could still be possible through various techniques, including timing attacks. In order to prevent from such attacks, numerous solutions have been proposed in the literature. In [Baran, 1964], the authors propose adding dummy traffic to hide the actual amount of the original traffic, while others [Syverson et al., 1997] suggest the use of constant rate padding between each hop to protect users anonymity. Finally, another approach [Fu et al., 2003] is using a random delay in each message in order to prevent the execution of timing or similar kind of attacks.

We implemented two different types of scenarios for the above scheme. In the first scenario, public-key encryption is used for field protection, based on SIP proxy server certificates, while in the second one symmetric-key encryption is used. We compared our solution with Tor, I2P, Orbot [Orbot, 2008], which is a smartphone proxy app allowing traffic redirection through Tor, and a previous approach of ours called PrivaSIP over Tor [Karopoulos et al., 2014] based on Tor network and PrivaSIP [Karopoulos et al., 2010, Karopoulos et al., 2011]. The experimental results presented in Section 5, show that PrivaSIP over Tor is less efficient than plain SIP over Tor by almost 1 sec. However, only the former protects users' IDs from intermediate SIP proxies; thus the trade-off here is more delay for more privacy. Overall, PrivaSIP over Tor is still affordable, since it takes around 2 secs for a call to be established. OnionSIP, on the other hand, presents better performance from related solutions, while at the same time offers more advanced privacy protection compared to PrivaSIP over Tor by not exposing communicating domains to intermediate proxies.

The rest of the paper is structured as follows. The next section presents related work, along with the hurdles we faced and other solutions that we rejected. In Section 3, we present our proposal of an onion routing inspired, privacy-preserving framework for SIP. Any implementation and technical details related to our experimental testbed are mentioned in Section 4, while in Section 5 we present the evaluation and the comparison of all schemes, in terms of delay and qualitative characteristics. The last section concludes the paper and gives some directions for further research.

2 Related work

This section presents related work on anonymizing networks that could potentially be used for anonymizing SIP. We start by analyzing schemes that are not suitable for SIP explaining the reasons, followed by a more detailed analysis of the major anonymizing networks that can be used for this purpose, i.e., Tor and I2P.

Most anonymity networks target diverse needs, so existing systems are implemented with a specific purpose in mind. First and foremost, the majority of the considered anonymity systems or tools rely on Tor or other widespread networks to achieve anonymity, wrapped and foisted with some minor changes.

Other systems, like iMule [iMule, 2003], StealthNet [StealthNet, 2007] or Mixmaster [Parekh, 1996], are used for different purposes such as sending emails or exchange files anonymously, without giving the choice of proxy for a third app like a SIP client.

There is also a plethora of networks proposed in the bibliography that promise anonymity, but they are on an early or prototype stage, such as Tarzan [Freedman and Morris, 2002], MorphMix [Rennhard and Plattner, 2002], and Phantom protocol [Bråding, 2011]. Hornet [Chen et al., 2015], which name is an acronym for High-speed Onion Routing at the Network Layer, is yet another anonymity network, similar to Tor. Their creators claim that Hornet will be able to reach speeds exceeding Tor network. However, there is still, by the time this paper was written, no tangible implementation of Hornet, and thus there is no way to test it. Last but not least, MIT researchers anounced that they have developed and will soon present, a new onion-routing based anonymity scheme called Riffle [Kwon et al., 2016]. They claim that Riffle provides strong security and anonymity using bandwidth much more efficiently than other anonymity networks. However, it is still unpublished and thus, we were unable to test it with SIP.

Other researchers used various approaches to anonymize or preserve privacy in VoIP systems. In [Karopoulos et al., 2014] we proposed a premature effort on using Tor to anonymize SIP traffic. We succeeded in providing full SIP message obfuscation by adapting an application-layer, encryption framework, called PrivaSIP [Karopoulos et al., 2010, Karopoulos et al., 2011]. Using PrivaSIP over Tor, we were able to encrypt the sensitive contents of SIP messages, including `From` and `To` header fields, thus preventing unauthorized users from reading these fields. The combination of PrivaSIP with Tor can be considered a complete SIP privacy solution, as it achieves both application and network layer privacy. Since PrivaSIP over Tor serves a similar purpose as OnionSIP, we have included it in the experimental results presented in Section 5.

In [Zhang and Fischer-Hübner, 2010], the authors propose a solution which is resistant against two attacks (complementary matching and watermark) that a malicious user can perform to discover “who calls whom”, even though traffic is proxyfied through an anonymization overlay network (AON). The authors claim that they found a way to prevent attackers performing such attacks, by applying Voice Activity Detection in every call. That is, the originator’s user-agent can produce voice packets in a constant rate and instruct the AON proxies to drop those packets, hindering the attacker to disclose users identities. Nevertheless, as the authors mention, the above defense is not a comprehensive solution that can provide highly usable, efficient and practical anonymity for all VoIP users.

In [Kambourakis, 2014], the author details the different aspects of anonymity and accountability in the SIP protocol. Moreover, he presents the two major categories that privacy mechanisms are divided into, the cryptographic-based solutions, like SIP over TLS and SIP using S/MIME, and the non-cryptographic ones, which anonymize the SIP URI. Based on this separation, a discussion is made about the custom solutions, plus the generic and lower layer ones. The protection of SIP using S/MIME is documented extensively in RFC 3853 [Peterson, 2004]. Each message is treated like an email attachment and so it is encrypted, using an algorithm like AES, and signed via S/MIME. Using such a methodology, all the security services that S/MIME provides can be added to SIP, including authentication, message integrity, non-repudiation of origin, privacy, and data security. However, in S/MIME, some of the header fields of the message must always remain in plaintext, as they are necessary for successful message delivery. More specifically, `To, From, Call-ID,` and `Contact` fields are still unprotected. The inability of S/MIME to protect all the sensitive fields of a SIP message, is a defect that OnionSIP achieves to solve. This is realised by encrypting each sensitive field individually. Moreover, OnionSIP is more flexible, as the type of encryption and which fields will be encrypted can be selected according to user's needs as the case may be.

Herd [Le Blond et al., 2015] is a scalable, traffic analysis resistant anonymity network designed for VoIP systems. The authors claim that Herd provides user anonymity by forwarding traffic through cloud-based proxies that can be considered as low-delay circuits. Herd uses a hop-by-hop and layered encryption over a circuit of nodes, using Datagram TLS (DTLS) to accomplish lower latency than other similar anonymity networks. The proposed solution is claimed by the authors to be quite efficient with lower bandwidth requirements, however it achieves that through a dedicated infrastructure, incurring additional costs and delays.

In [Heuser et al., 2017], the authors highlight the problem arising from the metadata contained in Call Detail Records (CDRs), which contain sensitive information such as source, destination, start time and duration of a call. They propose Phonion, an architecture where every call is routed over the telephony infrastructure. The authors claim that Phonion can achieve high quality calls while obfuscating call data records. However, Phonion only protects the CDRs and not the actual contents of the VoIP messages, thus, a call is still vulnerable to traffic analysis attacks.

2.1 Tor

The Onion Router (Tor) [Dingledine et al., 2004] is the most widely used anonymity network at the moment. Most users who need to preserve their anonymity and privacy on the Internet, turn to Tor as it consists of a large amount (currently almost 7,000) of nodes [TorStatus, 2017] from all over the world, offering multiple layers of encryption while obfuscating user IPs. One of Tor's biggest advantages is the ability to support a plethora of applications, including BitTorrent and HTTP.

Tor actually uses a large amount of volunteers participating in it, to create a decentralized anonymity network. Each volunteer takes the role of a node, and multiple nodes are setting up a circuit that intervenes between the sender and the receiver of the packet. To create this circuit, the user who wishes to surf anonymously, let's call him Bob, uses his Onion Proxy (OP) to collect all the possible nodes of the Tor network from the central server and chooses a number of them (usually three) which will be used. The server will send back to the client all the available, entry, middle and exit, nodes. Then, according to some set of rules, Bob's OP will choose the three nodes that will constitute his path to the final destination. In this point, we should mention that each onion router (OR) maintains a TLS connection to every other OR. Bob and the entry node negotiate a session key, which will be used for transferring packets securely as long as this connection stays alive. The above procedure is repeated for each node Bob needs to extend his circuit with.

Keeping a session key with each node and wrapping a message into each one of these session keys, secures the message into a multilayer encryption, just like an onion. None of the nodes participating in the circuit can read the actual contents of a packet, except the exit node which decrypts the last layer of encryption and forwards the message to the final destination.

2.2 Orbot

Orbot [Orbot, 2008] is a free anonymity and privacy-preserving software for Android. It uses Tor network as a proxy and allows traffic originating from a device's web browser to be routed through Tor, providing anonymity for the user. It also has the ability to transport through Tor all the TCP traffic on an Android device, given the correct permissions and system libraries.

2.3 I2P

Shortly after Tor's release, another open-source network appeared called Invisible Internet Project, also known as I2P [Zantout and Haraty, 2011], providing anonymity and security. It is notable, that I2P is based on garlic routing, contrary to Tor's onion. Garlic routing was discovered to emphasize general differences from Tor, meaning that multiple packets are encrypted and sent together, just like a garlic. This method of message encryption, was introduced trying to give an extra protection from traffic analysis. I2P also uses a multiple layered encryption, very similar to Tor, using each node's public key. Then each node decrypts one layer at a time and sends the packet to the next node, as instructed. Still, the only information each node has is the routing instructions.

In I2P, multiple messages, named cloves, are bundled together and then encrypted, while each message still carries its own routing instructions, which are exposed to the endpoint. However, this is not the only difference from Tor's architecture. I2P uses unidirectional tunnels, that is, there is no single path for sending and receiving packets. Each I2P party builds two different tunnels, one for the incoming and one for the outgoing traffic and therefore, four different tunnels are required for a single round-trip message and reply. Each tunnel uses its own session keys with the initiator, while encryption is based on ElGamal/AES.

2.4 Attacker model

Our proposal focuses on privacy protection of SIP signaling, thus, in the following attacker model we only consider privacy-related attacks. As communication model, we assume the traditional SIP trapezoid, where two users communicate using two intermediate SIP proxies. We identify two main classes of attackers:

SIP proxies. While SIP proxies are system entities, they cannot be completely trusted (e.g. a mobile user can utilise an outbound proxy that does not belong to their service provider). Here, we argue that SIP proxies follow the honest-but-curious model. According to this model, the proxies are assumed not to drop or modify messages routed through them, allowing the system to run smoothly, while at the same time try to infer private information from the exchanged messages. This way, the proxies, as well as the entities that control them, can mount traffic analysis attacks and get access to private user information like caller and callee usernames, IP addresses, and network domains.

Third parties. This class concerns external attackers that are neither system entities nor participate in the call. As SIP is a text based protocol and messages are sent in plaintext, third parties can eavesdrop on all messages that are exchanged among users and SIP proxies. These attackers can perform traffic analysis attacks and access caller and callee usernames, IP addresses, and network domains. The difference from the previous class is that third parties can also modify or drop SIP messages in order to mount Denial-of-Service or call hijacking.

3 OnionSIP

As already pointed out, SIP is a protocol whose messages are transferred in plain text, thus, any third party is able to identify the caller, the callee, as well as their locations. To protect these sensitive information, in OnionSIP we encrypt any fields that should be hidden from any parties that are not authorised to read. That is, any user or proxy should be able to read only the information needed to make the call establishment possible. Our proposal is a multilayered encryption scheme, similar to onion-routing, which gives the ability to every node to decrypt the appropriate fields and forward the message to the next node, based on the information it decrypted. The above idea is based on the fact that intermediate proxies in a SIP path are able to forward packets without problems, although unused sensitive information are hidden from their sight. Similarly to other anonymity networks, OnionSIP's degree of anonymity is highly dependent on the number of gateways that are used between each party. The more the gateways are, the higher the level of privacy is.

OnionSIP is a novel scheme which constitutes a more efficient solution than the previously proposed PrivaSIP over Tor [Karopoulos et al., 2010, Karopoulos et al., 2011], while at the same time preserving the high level of privacy and anonymity the later offered (i.e., protecting call metadata even from intermediate proxies). In contrast to PrivaSIP over Tor, OnionSIP does not use any third party anonymization system, like Tor. To achieve user anonymity and privacy, OnionSIP hides the content of each SIP message individually by encrypting the necessary fields and not the whole message.

To choose which fields are considered sensitive and should be protected, we first consider which ones contain useful information about the call or the users participating in it. First of all, the INVITE URI contains information about the final user (i.e., the username and the destination domain). In addition, `From` field contains information about the caller, while `To` field holds the "logical recipient" of the message, which may or may not be the ultimate recipient of the request. These fields include the username, or in some cases even the real name, of the user as well as the domain name of their Registar. Moreover, the `Call-ID` is used for giving uniqueness to a session. Apart from the fact that `Call-ID` could help an observer correlate two different SIP messages, it also includes the sender's IP address, making it possible for the observer to identify every node that is involved in the session. Last but not least, the `Contact` field is used to represent a direct route to contact the caller. All the above fields are considered sensitive, as they can easily expose information about the call and so they should be protected. For the proposed system, we apply two different encryption schemes: (a) an asymmetric, and (b) a symmetric one.

3.1 OnionSIP asymmetric

The first of the two variations of our proposal makes use of public-key encryption to encrypt the appropriate fields of SIP messages. It is assumed that the caller already possesses the public keys of all the hops in the path of the call towards the callee, including the callee's public key. To explain how our framework works we assume that Alice, registered to SIP Proxy A, wants to call Bob, who is registered to SIP Proxy B, as depicted in Figure 1. Sensitive fields in the INVITE message are being encrypted by Alice in layers in such a way that each party can decrypt one layer obtaining information for the next hop only. Below we present the steps that take place for the call establishment:

Step 1: Alice concatenates `INVITE, From, To, Call-ID` and `Contact` fields forming the Layer1 message shown in Figure 1. Then, she encrypts it with Bob's public key (KC).

Step 2: Alice appends the string "`To: Bob@proxyB.org`" to this encrypted message and encrypts the result with the public key of Proxy B (KB).

Step 3: Alice concatenates the string "`To: anon@proxyB.org`" to the last encrypted message and encrypts the result with Proxy's A public key (KA).

Step 4: The encrypted message is placed in the message body of the SIP message, while SIP header fields are all replaced with anonymous SIP URIs, for example "`To: anon@hidden.org`".

Step 5: Alice forwards the resulting SIP message to its own proxy, i.e., Proxy A.

Step 6: Proxy A decrypts the message it finds in the SIP message body using its private key (KA^{-1}).

Step 7: Proxy A replaces, in the SIP message it received, header `To` with the decrypted value, i.e., "`To: anon@proxyB.org`"; this way it knows the next hop which is Proxy B without knowing the final destination, i.e., Bob.

Step 8: Proxy A replaces the SIP message body with the decrypted message without including the "`To: anon@proxyB.org`" value.

Step 9: Before forwarding the INVITE message, Proxy A generates a TRYING message and encrypts any sensitive information contained in it, using Alice's public key and sends it back to her.

Step 10: In a similar way, Proxy B decrypts the message it finds in the message body, using its private key (KB^{-1}), replaces the `To` field with "`To: Bob@proxyB.org`", and also replaces the message body with the decrypted message, responding back to Proxy A with an onion-encrypted TRYING message.

Step 11: Finally, Bob decrypts the message it finds in the SIP message body with his own private key (KC^{-1}), where he can find all the necessary information about the caller.

Step 12: If Bob needs to send a reply, he follows the same concept and encrypts sensitive information using the public keys of the same path in the reverse order or a totally different path for enhanced privacy.

The above procedure is presented in Figure 1. As one can observe, the domain of the `To` header field in Msg 1 is not hidden, as it contains information about the next hop.

3.2 OnionSIP symmetric

In this variation, the framework works as mentioned above, with the main difference being that the parties that need to communicate with each other already share a symmetric key to encrypt SIP message contents. We assume that involved parties already have digital certificates and the caller has earlier established a different session key with each of the parties along the path to the callee, i.e., Alice should negotiate session keys with Proxy A, Proxy B and Bob during the registration process by means of, say, a Diffie-Hellman handshake. This implementation is close to how modern anonymization networks using onion routing (like Tor) work, as they use symmetric encryption to protect traffic traversing through each node of the chain.

3.3 Key exchange

The correct operation of OnionSIP requires that Alice and Bob exchange their public keys or agree to a common key. This kind of key exchange can lead to privacy leakage to an adversary that observes communicating IP addresses; however, apart from the direct exchange of keys, there are alternative solutions that do not breach user privacy. First of all, Alice can acquire Bob's certificate from an LDAP server or a Certificate Authority and vice versa. This is a common solution that can take place whenever two users need to communicate with each other for the first time. Moreover, when a user receives a certificate, the latter can be stored for future transactions, as long as the key is considered long-termed. In the case of symmetric keys, a Key Distribution Center can be used in order to avoid direct communication; these session keys can also be stored for a limited period and used for more than one call.

In our experimental results, we have not involved key exchange/agreement delays as a plethora of parameters and conditions could affect the key exchange phase in different ways. For instance, some keys can be stored during previous

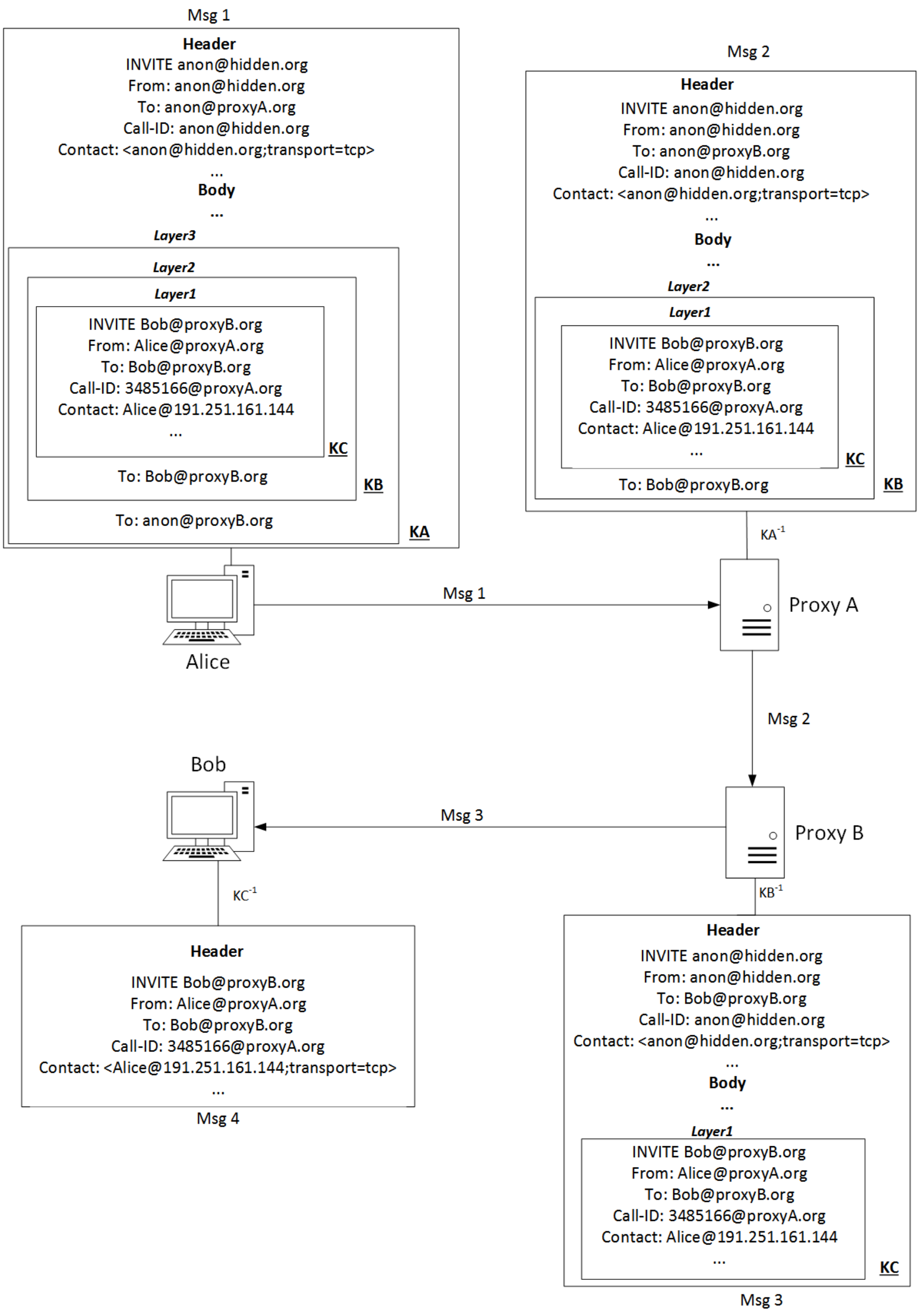

Figure 1: Example of anonymizing a SIP INVITE message

Scheme	Proxy A	Proxy B	Client A	Client B
OnionSIP-AES	Kamailio v.4.4.0	Kamailio v.4.4.0	Jitsi v.2.9	Jitsi v.2.9
OnionSIP-RSA	Kamailio v.4.4.0	Kamailio v.4.4.0	Jitsi v.2.9	Jitsi v.2.9
Tor	Kamailio v.4.4.0	Kamailio v.4.4.0	Twinkle v.1.4.2	Twinkle v.1.4.2
I2P	Kamailio v.4.4.0	Kamailio v.4.4.0	Twinkle v.1.4.2	Twinkle v.1.4.2
Orbot	Kamailio v.4.4.0	Kamailio v.4.4.0	CSipSimple v.1.02.03	Sipdroid v.3.0
PrivaSIP over Tor	SER v.0.9.6	SER v.0.9.6	Twinkle v.1.4.2	Twinkle v.1.4.2

Table 2: Software characteristics per node

sessions and reused in future ones. In addition, Tor's performance is highly dependent upon various parameters of each node that is part of the final circuit, like the location and the bandwidth of each node. As the authors in [Wendolsky et al., 2007] mention, it usually takes almost 4 seconds in average for a Tor client to establish a circuit. This is a significant delay, meaning that in a real world application an established circuit would be reused, thus, leaving the key agreement procedure out of the session establishment.

4 Implementation

For the first five candidate solutions, we utilised Kamailio Server v.4.4 on Cloud-based VM's hosted on our own infrastructure for SIP Proxies, while SER v.0.9.6 was used on the sixth one, given in [Karopoulos et al., 2014]. The Kamailio servers were installed on Ubuntu 14.04. Each VM, of both clients and servers, consisted of 4GB RAM, an Intel Xeon E5-2690 processor and 60GB of SSD storage. Each SIP client, behaves differently to each of the anonymization systems, so we chose different clients for each scenario, depending on which one worked. We chose to use Twinkle v1.4.2 for both Tor and I2P, while for Orbot we used CSipSimple on the caller's side and Sipdroid on the callee's one. Finally, for the OnionSIP-AES and OnionSIP-RSA schemes, we used Jitsi v2.9 for both caller and callee. The above software characteristics are summarised in Table 2, while Table 3 presents the hardware characteristics of each host.

In Table 4 we make a layered representation of the different platforms we examined in this paper. We present each system we used for the purpose of this

Machine	CPU	RAM	OS
Proxy A	Single-Core Intel Xeon E5-2690	4GB	Centos 7 kernel v.3.10
Proxy B	Single-Core Intel Xeon E5-2690	4GB	Centos 7 kernel v.3.10
Client A	Single-Core Intel Xeon E5-2690	4GB	Ubuntu 14.04 kernel v.3.16
Client B	Single-Core Intel Xeon E5-2690	4GB	Ubuntu 14.04 kernel v.3.16
Android Caller	Snapdragon 800 Quad-core 2.3 GHz	2GB	Android 6.0
Android Callee	Snapdragon 800 Quad-core 2.3 GHz	2GB	Android 5.0

Table 3: Hardware characteristics per node

OSI Layers	Platforms/Systems	Type
Application	Jitsi	SIP Client
	SipDroid	Android SIP Client
	CSipSimple	Android SIP Client
	Twinkle	SIP Client
	OnionSIP	Anonymity Network
Presentation		
Session	Tor	Anonymity Network
	Orbot	Anonymity Network
Transport		
Network	I2P	Anonymity Network
Data Link		
Physical		

Table 4: OSI layer placement of used platforms

paper and in which OSI layer it belongs, along with the type of each system. OnionSIP along with the two SIP clients are both sitting on the Application Layer. Tor and its mobile version Orbot are using a SOCKS proxy to forward traffic through the onion-routing network, so they both operate on the Session Layer [Bowne, 2009]. On the other hand, I2P uses its own API to anonymize traffic rather than a SOCKS proxy, so it is built on top of the Network Layer [I2P, 2013].

4.1 Tor

To anonymize traffic originating from a SIP client or a SIP Proxy, we should make use of Tor network as an intermediate, using Tor's default port 9050 on localhost. As there is no SIP client able to handle SOCKS5 connections, we used Proxychains [ProxyChains, 2002], a software which is able to tunnel traffic through any proxy server, in order to force Twinkle to use Tor. Proxychains advantage is its SOCKS5 support, which is the main protocol for transferring packets through Tor. On the other hand, we used `iptables` routing rules to force any traffic coming from the caller's proxy to get through the Tor network, through Tor's port 9050. However, this is necessary only on the caller's proxy side while communicating with the callee's proxy. The caller and the callee have already established a connection through Tor with their proxies during Register.

4.2 Orbot

For testing Orbot, we used two rooted Android smartphones, a Nexus 5 with Qualcomm MSM8974 Snapdragon 800 processor and 2GB of RAM as a caller, and an LG G3, with a Qualcomm MSM8974AC Snapdragon 801 processor and 3GB of RAM as a callee. We also used CSipSimple as the caller and set up Orbot to anonymize any traffic coming from it, while the callee used Sipdroid.

4.3 I2P

I2P administrators, have cut down SOCKS outproxies support, making it impossible to create and use a SOCKS proxy as an exit-node to the Internet. The only alternative is to use Tor as an exit node, but this is considered already a heavy burden to carry. However, considering the "HTMLish" form of SIP, it is possible to use HTTP proxies that are provided for anonymity networks, like I2P. The clients are using Proxychains to connect to I2P network through I2P's default port (4444) on localhost. Additionally, the caller's proxy is using `iptables` routing rules to redirect any outgoing traffic through port 4444. Nevertheless, if two users want to communicate securely, they both need to be part of the I2P network. If they are not, an outproxy needs to be used, transferring traffic from I2P to the Internet. In that case, the outproxy acts similarly to a Tor exit node, thus after that step all traffic is in its original form (i.e., if no encryption was initially used then it is in plaintext).

4.4 OnionSIP

As described above, for OnionSIP, we implemented two different schemes, with the first using asymmetric encryption to achieve onion-routing, while the other

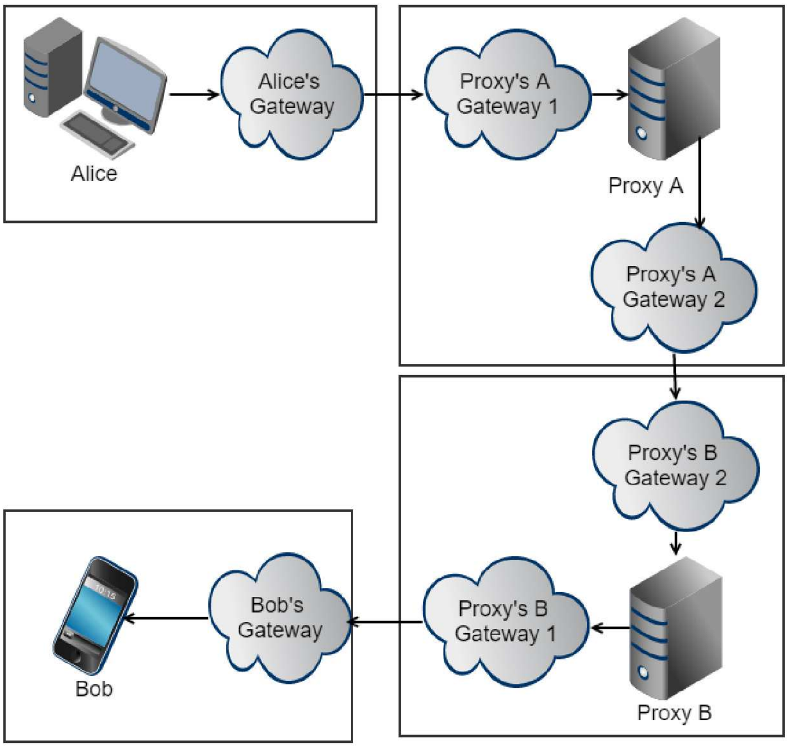

Figure 2: Architecture of OnionSIP multi-layer encryption

is using symmetric encryption. For the first version, we chose RSA as the public-key algorithm, with each key having a 2048 bit size. The symmetric-key of the later is implemented using AES with 128 bit long symmetric keys, similarly to Tor [Dingledine et al., 2004].

We implemented six different SOCKS gateways written in Python, which intervene between the communicating parties as shown in Figure 2. The caller and the callee have one gateway proxy each, serving both for encryption and decryption. Each SIP proxy has two gateways: the first is used for decrypting incoming traffic, while the second one is used for encrypting outgoing traffic. For example, Proxy's B Gateway 1, is responsible for encrypting and decrypting any traffic that is sent to or received by Bob, while Gateway 2 is responsible for handling communication between Proxy B and Proxy A. The gateways may coexist with the proxies and/or the end-user machine or operate in different machines. Each party communicates using a TCP connection, as SOCKS5 requires applications to establish a TCP connection to the SOCKS proxy before it forwards any packet.

It is worth noting that using such a scheme, SIP messages consist of extra, unknown information in their body, without affecting the call establishment in any way. This means that OnionSIP is fully compatible with SIP and can be used without modifications to SIP infrastructure except the addition of the corresponding gateways. Kamailio does not use any SDP parser to check if the messages transferred are valid or malicious in any way. This can constitute a major security flaw for a widely-used SIP proxy, like Kamailio, and thus it should be taken into account before wide deployment of this solution [Tsiatsikas et al., 2015].

5 Evaluation

For the evaluation and comparison of the aforementioned systems, we measured the establishment delay of at least 100 SIP calls for each scheme. First, Alice, who acts as a caller, sends an INVITE message to Bob, who accordingly acts as a callee. Alice receives a 100 (TRYING) message while the proxies, which intervene between the call, try to forward the request to Bob. Finally, when Bob receives the INVITE message, he replies to Alice through the involved proxies with a 180 (RINGING) message. We measure the time between the moment Alice sends the INVITE message and the time she receives the RINGING message back from Bob. Following the above procedure, we produced more than 100 calls sequentially with the help of SIPp tool. We utilized Wireshark on the caller's side and we were able to compute each call establishment delay. The derived values were rounded to one decimal digit and the frequency per value was counted.

In Figure 3, we present the comparison of the six different alternatives which attempt to preserve user privacy in SIP, using the Cumulative Distribution Function (CDF). OnionSIP based on AES is the most efficient scheme, according to our results, followed by OnionSIP based on RSA, which has a slightly better performance than Tor. In order to have comparable results with Tor, in our AES variation experiments we do not take into account the session key establishment delays. This operation, just like in Tor, takes place one time and, after that, the call establishment delays are those shown in Figure 3. Tor has close performance with Orbot, something that was expected since Orbot is based on Tor; they both show delays mainly between 0.9 and 1.1 seconds. I2P presents delays between 1.1 and 1.5 seconds; this result was expected since SOCKS proxies are by definition noticeable faster that HTTP ones. Finally, PrivaSIP over Tor [Karopoulos et al., 2014], is the least efficient scheme, in terms of performance from the rest of the schemes.

Taking into consideration the results produced by the CDF graph, one can observe that PrivaSIP-over-Tor has a narrow range between 1.7 and 2.1 seconds, while other schemes have a wider range, approximately between 0.3 and 1.5.

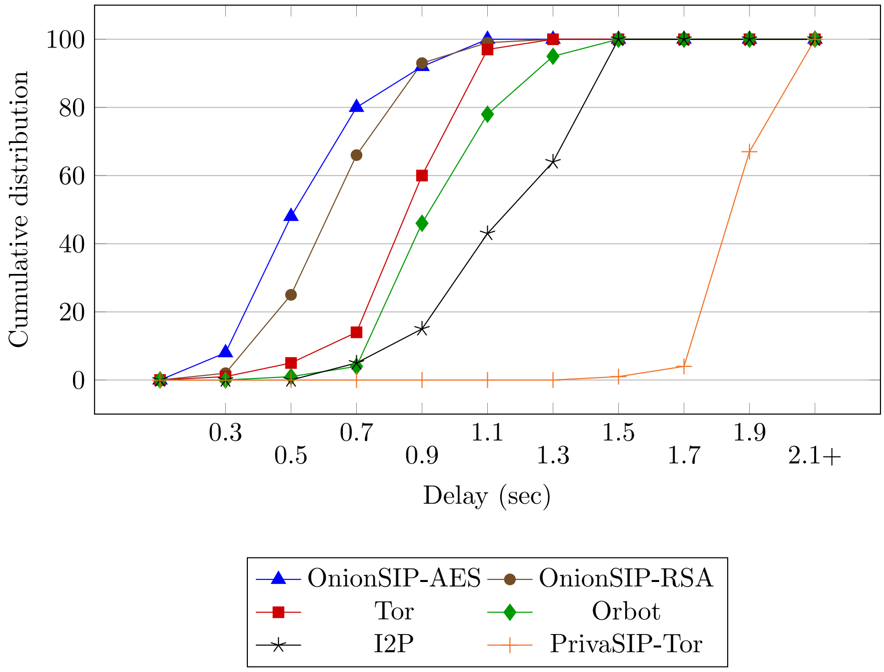

Figure 3: Delay comparison of privacy schemes for SIP

Also, Orbot and PrivaSIP-over-Tor present larger delays compared to Tor as expected, since they are based on Tor adding other mechanisms on top of it. In Table 5, we introduce the mean and standard deviation metrics for each solution. In this table, one can observe that both OnionSIP and I2P solutions produce the lowest standard deviation compared to the rest of the proposals. Bear in mind that the lower the standard deviation, the more data are clustered around the mean value. So, despite the fact that PrivaSIP-over-Tor seems to have a narrower range (1.7-2.1 secs), OnionSIP introduces more predictable delays; this observation holds both for OnionSIP based on AES as well as on RSA. Even if the two flavours of OnionSIP are based on symmetric and asymmetric cryptography respectively, we chose to compare them so that adopters can decide whether to take the extra burden required for key management in the symmetric case or not.

In the box-and-whisker plot presented in Figure 4, the statistical distribution of the call establishment delays is shown. In this plot, the median, the interquartile range, and the range are represented. This plot confirms that OnionSIP, in both of its variations, is more efficient than other privacy preserving schemes.

Table 6 presents a brief comparison of the evaluated schemes indicating the most important criteria each one satisfies. The first two criteria are related to

Solutions	Mean	Standard Deviation
OnionSIP-AES	0.6528	14.514
OnionSIP-RSA	0.7265	14.8324
Tor	0.9505	16.970
Orbot	1.0508	15.318
I2P	1.2486	13.523
PrivaSIP over Tor	1.9582	21.260

Table 5: Comparison of each solution's standard deviation

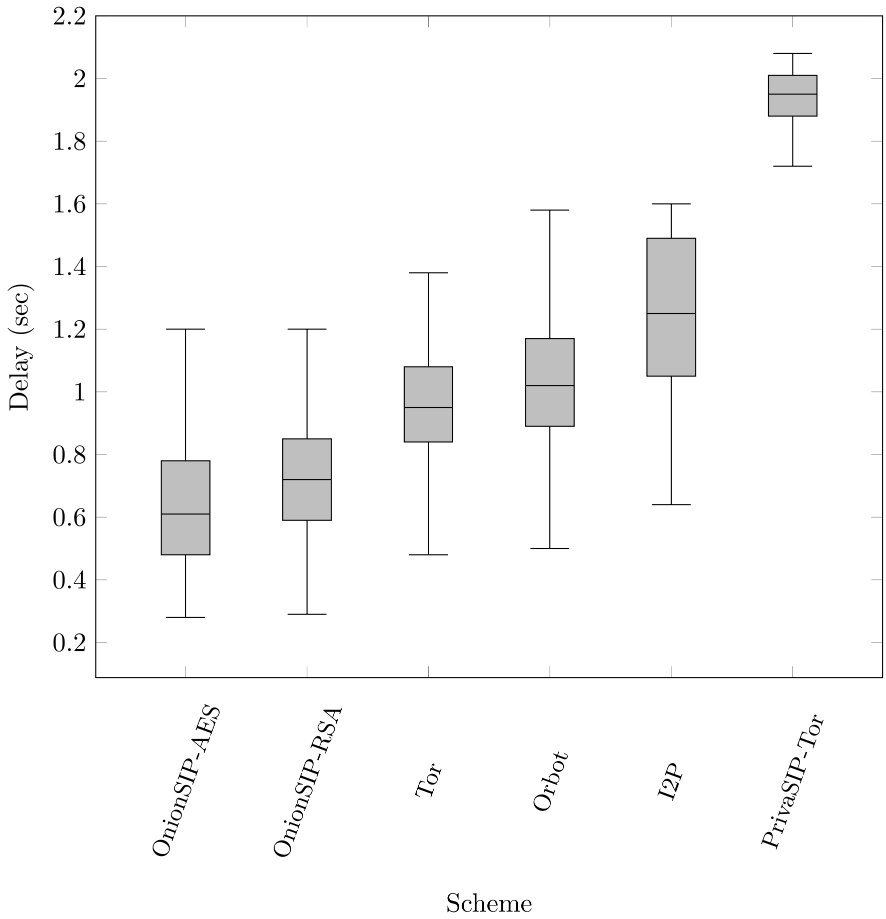

Figure 4: Box-and-Whisker plots representation of SIP call establishment delays

Anonymity and Privacy, which all the schemes satisfy. The third one concerns the type of message encryption each scheme uses. For instance, while OnionSIP encrypts only the sensitive fields of a SIP message, Tor and I2P use a different method by encrypting the whole message. The type of encryption each scheme uses, affects the way each proxy stores its logs. So, while both OnionSIP and PrivaSIP encrypt only selected SIP message fields, the intermediate parties will finally store logs where the sensitive information of each message is hidden. On the other hand, in Tor and I2P, the SIP proxies will store logs with each message having its initial form. According to the above, while all solutions offer privacy against third parties, only OnionSIP and PrivaSIP obfuscate user identity from intermediate SIP proxies. Furthermore, as all the schemes use some kind of encryption, we should distinguish which of them need extra time to establish any keys needed; apart from OnionSIP-RSA, the rest of the schemes will need extra time for key establishment, circuit creation, etc. Another important criterion is which scheme needs an extra application or system configuration, in order to function properly. In that case, schemes using Tor or I2P need a proxifier to redirect any traffic through them; OnionSIP variations work on the go and thus, no external application is needed. None of these solutions are based on any single point or central server, so all solutions offer decentralization. When it comes to the maturity of the codebase, only Tor offers a mature solution, as I2P is an anonymization network which requires many improvements and the rest of the solutions are in proof-of-concept status. Finally, based on the fact that I2P requires both users to be part of I2P network or specify an outproxy in combination with the fact that an external application is needed, makes the use of I2P a rather difficult task. Additionally, Tor and PrivaSIP over Tor, require the use of an external software to redirect traffic through Tor network. Considering all these facts, OnionSIP forms a privacy solution which could be characterized by ease of deployment.

6 Conclusion

When two users need to communicate and various servers and proxies intervene to establish their connection, any malicious part can potentially capture sensitive information about the communicating users. User identity disclosure is one of the most common problems that users participating in a VOIP call face. When it comes to SIP, there is a limited number of solutions to prevent a malicious actor to identify the communicating end-users. An easily deployable solution would be the modification of existing anonymity networks, which are intended for other applications like the `www`, in order to support SIP calls. However, at least up to now, most anonymizing networks do not support SOCKS or HTTP proxying, thus, they fail to anonymize SIP traffic, with Tor and I2P constituting an exception.

Criteria	OnionSIP-AES	OnionSIP-RSA	Tor	I2P	PrivaSIP over Tor
Anonymity	✓	✓	✓	✓	✓
Privacy	✓	✓	✓	✓	✓
Encryption	Certain fields	Certain fields	Whole message	Whole message	Certain fields
Privacy against proxies	✓	✓	✗	✗	✓
Privacy against third parties	✓	✓	✓	✓	✓
No extra time for key establishment	✗	✓	✗	✗	✗
External application independent	✓	✓	✗	✗	✗
Decentralization	3	3	3	3	3
Maturity of the codebase	1	1	3	2	1
Ease of Deployment	3	3	2	1	2

Table 6: Schemes comparison (✓: included, ✗: not included, 1: low, 2: medium, 3: high)

Another way to protect sensitive information included in SIP messages, is to encrypt the sensitive fields of each message in a multilayer fashion, like in onion-routing systems. In such a scheme, every authorized party will use its key (symmetric or asymmetric) to decrypt the top layer of the onion and acquire the appropriate information needed to forward the message to the next hop. The last node will decrypt the last layer, acquiring the original message that the initiator created. We proved that such a scheme offers an alternative solution for application-layer privacy with less delay than lower layer schemes, like Tor.

Anonymization networks, like Tor and I2P, may be considered a complete and easily deployable solution for preserving user privacy and anonymity, also offering protection from a plethora of other attacks. On the other hand, the use of OnionSIP, where selected sensitive fields are encrypted in a multilayer fashion, leads to lower delays while offering increased level of user privacy and anonymity.

References

[Baran, 1964] Baran, P. (1964). *On Distributed Communications: IX.: Security, Secrecy, and Tamper-free Considerations*. Rand Corporation.

[Bowne, 2009] Bowne, S. (2009). How socks5 works. `https://samsclass.info/122/proj/how-socks5-works.html`. Accessed: 2016-09-30.

[Bråding, 2011] Bråding, M. (2011). Generic, decentralized, unstoppable anonymity: the phantom protocol. *White paper*.

[Chen et al., 2015] Chen, C., Asoni, D. E., Barrera, D., Danezis, G., and Perrig, A. (2015). Hornet: High-speed onion routing at the network layer. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1441–1454. ACM.

[Dingledine et al., 2004] Dingledine, R., Mathewson, N., and Syverson, P. (2004). Tor: The second-generation onion router. Technical report, DTIC Document.

[Freedman and Morris, 2002] Freedman, M. J. and Morris, R. (2002). Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 193–206. ACM.

[Fu et al., 2003] Fu, X., Graham, B., Bettati, R., and Zhao, W. (2003). Active traffic analysis attacks and countermeasures. In *Computer Networks and Mobile Computing, 2003. ICCNMC 2003. 2003 International Conference on*, pages 31–39. IEEE.

[Heuser et al., 2017] Heuser, S., Reaves, B., Pendyala, P. K., Carter, H., Dmitrienko, A., Enck, W., Kiyavash, N., Sadeghi, A.-R., and Traynor, P. (2017). Phonion: Practical protection of metadata in telephony networks. *Proceedings on Privacy Enhancing Technologies*, 1:170–187.

[I2P, 2013] I2P (2013). The invisible internet project. `https://geti2p.net/en/`. Accessed: 2016-09-30.

[iMule, 2003] iMule (2003). `http://echelon.i2p.xyz/imule/`. Accessed: 2016-09-30.

[Kambourakis, 2014] Kambourakis, G. (2014). Anonymity and closely related terms in the cyberspace: An analysis by example. *Journal of information security and applications*, 19(1):2–17.

[Karopoulos et al., 2014] Karopoulos, G., Fakis, A., and Kambourakis, G. (2014). Complete sip message obfuscation: Privasip over tor. In *Availability, Reliability and Security (ARES), 2014 Ninth International Conference on*, pages 217–226. IEEE.

[Karopoulos et al., 2011] Karopoulos, G., Kambourakis, G., and Gritzalis, S. (2011). Privasip: ad-hoc identity privacy in sip. *Computer Standards & Interfaces*, 33(3):301–314.

[Karopoulos et al., 2010] Karopoulos, G., Kambourakis, G., Gritzalis, S., and Konstantinou, E. (2010). A framework for identity privacy in sip. *Journal of Network and Computer Applications*, 33(1):16–28.

[Kwon et al., 2016] Kwon, A., Lazar, D., Devadas, S., and Ford, B. (2016). Riffle. *Proceedings on Privacy Enhancing Technologies*, 2016(2):115–134.

[Kwon, 2015] Kwon, Y. H. (2015). *Riffle: An efficient communication system with strong anonymity*. PhD thesis, Massachusetts Institute of Technology.

[Le Blond et al., 2015] Le Blond, S., Choffnes, D., Caldwell, W., Druschel, P., and Merritt, N. (2015). Herd A scalable, traffic analysis resistant anonymity network for voip systems. In *ACM SIGCOMM Computer Communication Review*, volume 45, pages 639–652. ACM.

[Orbot, 2008] Orbot (2008). Tor on android. `https://www.torproject.org/docs/android.html.en`. Accessed: 2016-09-30.

[Parekh, 1996] Parekh, S. (1996). Prospects for remailers. *First Monday*, 1(2).

[Peterson, 2004] Peterson, J. (2004). S/MIME Advanced Encryption Standard (AES) Requirement for the Session Initiation Protocol (SIP). RFC 3853 (Proposed Standard).

[ProxyChains, 2002] ProxyChains (2002). Tcp and dns through proxy server. `http://proxychains.sourceforge.net/`. Accessed: 2016-09-30.

[Rennhard and Plattner, 2002] Rennhard, M. and Plattner, B. (2002). Introducing morphmix: peer-to-peer based anonymous internet usage with collusion detection. In *Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*, pages 91–102. ACM.

[Rosenberg et al., 2010] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and Schooler, E. (2010). Rfc 3261: Sip: Session initiation protocol.

[Salama et al., 2009] Salama, G. I., Shehab, M. E., Hafez, A., and Zaki, M. (2009). Performance analysis of transmitting voice over communication links implementing ipsec. In *Paper in 13th International Conference on Aerospace Sciences and Aviation Technology (ASAT), Military Technical College, Cairo, Egypt.*

[StealthNet, 2007] StealthNet (2007). `http://www.stealthnet.de/en_index.php`. Accessed: 2016-09-30.

[Syverson et al., 2001] Syverson, P., Tsudik, G., Reed, M., and Landwehr, C. (2001). Towards an analysis of onion routing security. In *Designing Privacy Enhancing Technologies*, pages 96–114. Springer.

[Syverson et al., 1997] Syverson, P. F., Goldschlag, D. M., and Reed, M. G. (1997). Anonymous connections and onion routing. In *Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on*, pages 44–54. IEEE.

[TorStatus, 2017] TorStatus (2017). `http://torstatus.blutmagie.de/`. Accessed: 2017-09-17.

[Tsiatsikas et al., 2015] Tsiatsikas, Z., Anagnostopoulos, M., Kambourakis, G., Lambrou, S., and Geneiatakis, D. (2015). Hidden in plain sight. sdp-based covert channel for botnet communication. In *International Conference on Trust and Privacy in Digital Business*, pages 48–59. Springer.

[Wendolsky et al., 2007] Wendolsky, R., Herrmann, D., and Federrath, H. (2007). Performance comparison of low-latency anonymisation services from a user perspective. In *International Workshop on Privacy Enhancing Technologies*, pages 233–253. Springer.

[Zantout and Haraty, 2011] Zantout, B. and Haraty, R. (2011). I2p data communication system. In *Proceedings of ICN*, pages 401–409.

[Zhang and Fischer-Hübner, 2010] Zhang, G. and Fischer-Hübner, S. (2010). Peer-to-peer voip communications using anonymisation overlay networks. In *IFIP International Conference on Communications and Multimedia Security*, pages 130–141. Springer.