Noname manuscript No.
(will be inserted by the editor)

AnonChat: A novel approach to a secure, anonymous and decentralized instant messaging

Babatunde Olabenjo

Received: date / Accepted: date

Abstract It is widely known that message encryption can improve the security of instant messaging applications. However, privacy becomes a concern when instant messaging users are trying to protect themselves against surveillance that threatens their personal freedom, confidential business activities and physical security. Most instant messaging applications provide little protection against eavesdropping, and a many do not offer a completely decentralized, anonymous and secure instant messaging platform. A centralized server architecture used in many instant messaging applications, still poses a security and privacy risk to users and providers of instant messaging services even if all communication is encrypted.

In this paper, firstly, modern secure communication and secure messaging methods were examined. Then, the architecture and security features of several current instant messaging applications were analyzed and reviewed, and scenarios were created to prove their weaknesses. Furthermore, the deficiencies of the examined systems revealed through these scenarios were considered to provide a theoretical basis to develop a decentralized, secure and anonymous novel instant messaging application called *AnonChat*. Finally, peer discovery and authentication times of both *AnonChat* and one of the existing instant messaging applications with similar features were analyzed in different geographical locations and network conditions. Also, comparisons were made between these two systems in order to evaluate the performance of *AnonChat*.

Keywords Instant Messaging · Internet · Cryptography · Data Security · Privacy · Tor · Anonymity Networks

Address(es) of author(s) should be given

1 Introduction

With the widespread of large-scale surveillance, including data mining and behaviorally targeted advertisement, the need to secure ones privacy and online communication becomes a necessity. With the recent outburst of large scale surveillance performed by the National Security Agency (NSA), and autocratic governments preventing their citizens from accessing certain information online, more users are taking privacy and security more seriously. The need for security arises when people need to send information over a medium without a third party intercepting their communication. Although, most cryptographic systems can provide conversation security, they are still inadequate to ensure the anonymity of the users.

Instant messaging systems mostly use server proxy and server broker architectures to transmit the messages to clients. Both of these architectures need a central server for operation. In the server proxy architecture, all instant messaging communication is passed through the server, and this is the default method that almost all major instant messaging networks use today. In the server-broker architecture, the only packets that are sent to the server are packets requesting the server to initiate communication between two clients. This method reduces the load on the server and reduces the privacy risk, as potentially confidential messages are no longer sent to the central server [17]. These systems that use server broker architecture require an authentication from a server or pool of servers and a directory list of clients in order to locate them. Most rendezvous protocols like Sun JXTA, SIP, Freenet Project and I2P, need Network Address Translation (NAT) transversal techniques, such as STUN in operation. [25]. Additionally, these protocols require an unblocked node or server

/ peer with a public IP address in order to communicate with the other peers in the network.

There are few systems providing anonymity and decentralization. Anonymous protocols such as I2P and Tor have been widely used by individuals to protect their privacy online [16]. For example, individuals who want to avoid tracking by advertising companies, or individuals who require access to websites and services blocked by their Internet Service Providers use Tor. Users in countries without freedom of speech and high censorship also use Tor to express themselves and send or access information from the Internet [12].

In this study, *AnonChat* is developed to approach the problem of decentralized, anonymous and secure peer-to-peer instant messaging system using existing protocols and tools. While considering existing secure systems, relevant systems that provide anonymity and privacy were considered. In our solution, no central infrastructure is required, single node failure will not hinder others in the network from communicating. Also, perfect forward secrecy is also included along with decentralization and anonymity. With this approach, previous messages cannot be decrypted by an intruder even if the encryption key is stolen. These distinctive features bring *AnonChat* one step ahead among others for secure instant messaging.

This paper is organized as follows. Section 2 introduces and explains communication protocols, current instant messaging systems, anonymity networks and their weaknesses. Section 3 provides the implementation of our proposed instant messaging application *AnonChat*. In addition, the section outlines *AnonChat*'s protocol and security. We also outline the features required in fulfilling the goal of this research. Section 4 discusses several performance tests with a comparison to a similar application in order to determine how the performance of our novel prototype is affected by the added security features. Finally, Section 5 summarizes our implementation and performance results and enhances further research direction.

2 Related Works

Rendezvous server models are used for initial bootstrapping in the current systems, such as Skype, Napster, and BitTorrent [30]. These protocols require a centralized server, where a request is made to locate all other existing peers/nodes on the network. Clients can be functional even when they have been located behind firewall because they can made all data exchange through the centralized server. However, if central server or system fails, this causes the collapse of the entire system. [30]. This section reviews similar communication protocols used in our prototype and current systems providing secure communication.

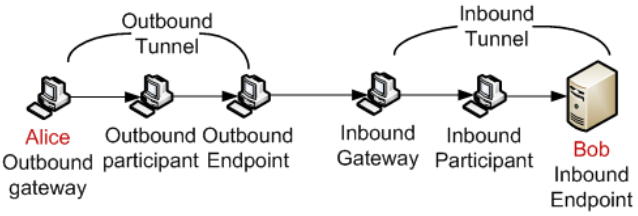

Fig. 1 I2P inbound and outbound tunnels

2.1 Communication and Security Protocols

In this section, some common communication and security protocols will be discussed. A few of these protocols combine both communication and security features while others are messaging protocols providing security such as Off-The-Record protocol (OTR).

2.1.1 Invisible Internet Project (I2P)

I2P formerly called Invisible Internet Protocol was a modification of Freenet; it was designed to be a scalable framework for anonymous communication [18]. I2P started in 2003, to enable anonymous communication in a dynamic and decentralized network environment that is resilient to attacks [12]. Figure 1 shows the two types of tunnels existing in I2P [15].

Several combinations of cryptographic methods are used in I2P to provide defenses against several forms of attacks. The network is mostly vulnerable on the application layer due to applications exposing their real IP and local information through meta-tags and Java Scripts. The location of a particular node can be demystified via statistical analysis [11].

2.1.2 Tor

Tor was designed to provide a general purpose and low-latency anonymity network stack based on a client-server architecture model [2]. It uses onion routing to move packets down the network where each node in the network only knows its predecessor and successor [8]. Encryption is done by using symmetric keys at each node and clients connect through a TCP tunnel that connects to randomly selected Tor routers [3]. Each Tor circuit consists of an entry guard, a middle router, and an exit node. Figure 2 describes the Tor network. To discover a router, the Tor client queries the directory server, which provides the address and public keys of the router.

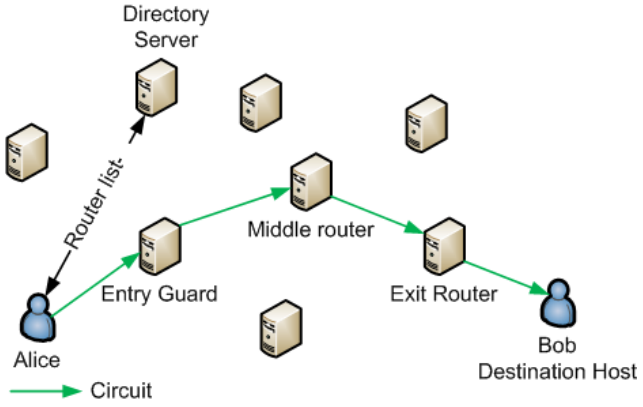

Fig. 2 The Tor network

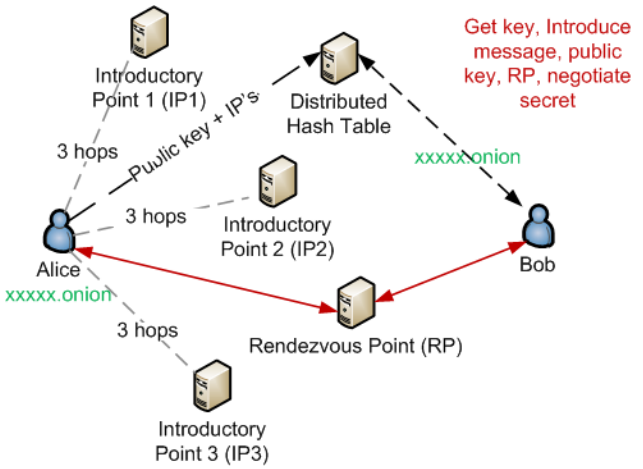

Fig. 3 Tor Hidden Service

Tor supports responder anonymity through hidden service. Using Tor hidden services, users can hide their location while providing various services such as instant messaging, web publishing and FTP access. Tor users can use services provided by others without knowing the servers network identity [23]. For a hidden service to function, user host needs to advertise itself to the Tor network. This is done by picking a set of relays in a random fashion. Next, user host builds a circuit and sends a request to relays with its public key to act as introductory points. The hidden service uses this information (introduction points, and public keys) to a distributed hash table. This is described in Figure 3. Tor generates an RSA-1024 key pair, and the SHA1 hash of the public key. This is used to generate the .onion address with exactly 16 characters.

2.1.3 Off-The-Record Messaging (OTR)

Off The Record Messaging, (OTR) is a cryptographic protocol that is designed for instant messaging. Encryption keys are created and destroyed during message exchange. Further details of its encryption mechanism are discussed in Section 3.3.5. OTR provides conversation over Instant Messaging (IM) with properties comparable to face-to-face conversation such that both users are assured of whom they are conversing with [28]. In OTR, future key compromises will not reveal past communication due to its perfect forward secrecy properties, short-lived encryption keys are used in encryption and keys are discarded after use.

2.1.4 Socialist Millionaire Protocol (SMP)

Currently, finding an excellent protection against Man in the Middle (MITM) attacks without using the Public Key and checking fingerprints have been shown to be difficult. OTR utilizes SMP to allow two parties check their secret are equal without revealing any other information to anyone else. SMP is derived from Yaos original millionaire problem. In this problem, two millionaires need to know who is richer without revealing their wealth [1]. MITM attack is prevented since the attacker cannot interfere with the SMP exchange. If there is an MITM, no information is leaked out to the middleman, and thus authentication will fail [6]. This method makes authentication seamless by providing an alternative means of authentication rather than the traditional fingerprint comparison. Details about the implementation of this protocol are discussed in Section 3.3.4.

2.2 Messaging Systems

In this section, related instant messaging systems such as Pidgin, Skype, TextSecure, TorChat, and CryptoCat are reviewed; their security features and architecture are also discussed and analyzed.

2.2.1 Pidgin

Pidgin is a multipurpose instant messenger application based on the libpurple library. Libpurple supports commonly used messaging protocols allowing users log into multiple services at the same time [20]. Pidgin supports multiple protocols, provides the extensible plug-in architecture to extend the application and its independent user interface, which separates the user interface from the functionality enables interoperability of various user interfaces. Since most protocols in Pidgin involve server-client architecture, messages traveling through the network to the server can be read, intercepted and stored. Any direct access to the server can expose conversations, locations, and other detail.

2.2.2 Skype

Skype is a peer-to-peer communication application that features both centralized and peer-to-peer communication; it is proprietary software owned by Microsoft. The central server is used for authentication and clients behind firewall or NAT are connected via STUN [24]. The connection is encrypted using AES-256 keys and RSA-2048. Although Skype provides strong encryption, a central server and closed source application is a potential threat to the privacy of its users. Skype consists primarily of Super Nodes, Ordinary Nodes, and Login Servers. Skype uses several cryptographic methods to protect data and authenticate peers on the network. It operates a certificate authority of names and authorizations by creating digital signatures for nodes thus verifying the identities of peers on the network [4]. Because Skype is proprietary, and none of its inner workings are open to evaluation, Skype does not portray itself as a fully secure application.

2.2.3 TextSecure

TextSecure is a messaging application that was developed by WhisperSystems to provide end-to-end encryption of messages [14]. TextSecure is open source and available on mobile devices. It encrypts messages on a users device and uses end-to-end encryption to transfer the encrypted message to other application users. TextSecure uses a variant of OTR (see Section 2.1.3); it improves this by using an asynchronous system of exchanging keys. Since OTR is synchronous in nature, clients need to wait for delivery of authentication messages before sending theirs. TextSecure provides asynchronous communication while providing perfect forward secrecy by using a different approach.

2.2.4 TorChat

TorChat is a decentralized anonymous instant messaging system using Tor as the underlying network. It provides real-time messaging using Tor hidden service. Rendezvous points in Tor network are the backbone of hidden services which TorChat uses. Some of the main security features of TorChat are its decentralized nature and anonymity. Messages traveling between peers are encrypted on the network layer, but the confidentiality of the message cannot be guaranteed [21]. TorChat does not authenticate its users but uses the Tor hidden service onion address that is an SHA-1 hash of the public key when the .onion address is created [13]. TorChat does not encrypt messages sent, and a compromised client can easily allow messages to be sniffed and read in clear text.

2.2.5 CryptoCat

CryptoCat is a free, open source instant messaging application that provides easy to use and encrypted instant messaging via a web browser extension [19]. CryptoCat aims to provide an easy to use encrypted messaging application, using OTR as the underlying encryption mechanism. Because CryptoCat is web-based, encrypted chat is easily accessible across several platforms. Several implementation of CryptoCat can enable encryption of instant messages over proprietary chat application online e.g. Facebook Chat. The client side is a browser extension that uses a JavaScript based implementation of OTR for one-to-one communication and multi-party OTR (mpOTR) for a multi-party communication.

3 Design and Implementation

While Tor network provides anonymity, there are various security and usability issues that may occur when trying to implement a secure messaging peer-to-peer anonymous system over Tor. Some of the issues on the underlying network are providing security against end-to-end attacks, protocol normalization and concealing who is connected to the Tor network [8]. Providing a desecrate anonymous instant messaging system where the sender and receiver do not know too much about each other which is a challenge. Currently, major implementations of Tor hidden services for secure communication and anonymity do not provide application layer message integrity. Also, if the private key of a hidden service is stolen, an intruder can impersonate another user. The following scenarios below describe the methods used by these applications to secure communication and their weakness. Our approach is to design a system *AnonChat*; that provides full discreet communication between individuals while maintaining the security and privacy of those individuals.

3.1 Scenarios

These scenarios describe several methods used to secure communication and some issues with these methods.

Scenario 1: In this scenario, Alice and Bob are communicating securely using a shared secret and a centralized server. Figure 4 represents details of scenario 1. In this situation, a third party Eve knows that Alice is talking to Bob. Also, the conversation can be decrypted if the secret key is found. Furthermore, the logs of their conversation can be retrieved directly from the server and analyzed.

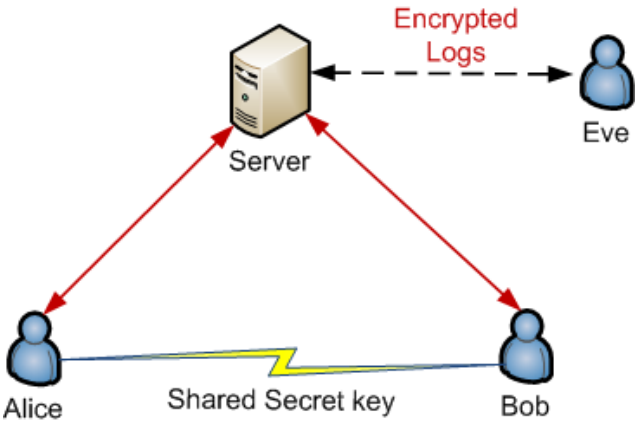

Fig. 4 Scenario 1

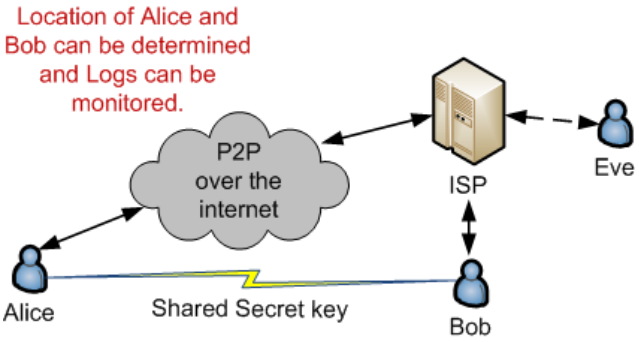

Fig. 5 Scenario 2

Scenario 2: Here, Alice and Bob decided to communicate in a peer-to-peer fashion in order to prevent logging on a centralized server. Figure 5 depicts details of Scenario 2. The Internet Service Provider can monitor communication using this method. Although, messages sent between Alice and Bob cannot be decrypted without the shared secret, the intruder Eve knows the data is going to Bob. Bob can be asked to give up encryption key to decrypt messages if interrogated.

Scenario 3: In order to hide their identities, Alice and Bob decided to use Tor to route their traffic as described in Figure 6. Here, Trudy does not know Alice is talking to Bob and the message still encrypted but if Bobs computer is stolen, Trudy can pretend he is Bob and get sensitive information from Alice. Also, if Trudy has been recording all previous conversations on Bobs exit nodes, the messages can now be decrypted with Bobs private keys.

What went wrong?

- In scenarios 1 & 2, Bobs key can be retrieved because they know Alice is talking to Bob.
- In scenario 3, no one knows whom Bob is talking to, but stealing Bobs data compromises the message. The message can still be decrypted including the previous messages received.
- All previous messages can be decrypted.

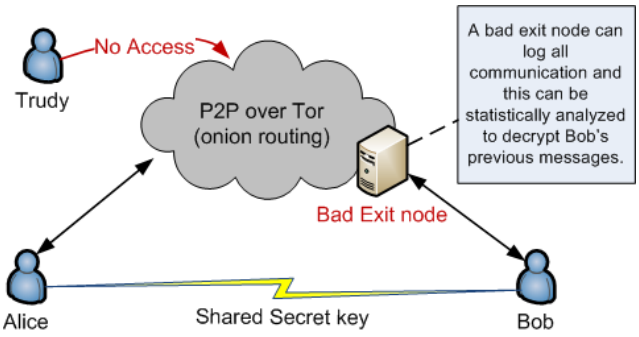

Fig. 6 Scenario 3

3.2 Solution Approach

In a recent survey done by the Electronic Frontier Foundation (EFF), several instant messaging applications were rated to determine how secure they were. This was called the Secure Messaging Scorecard. The majority of recent applications failed the test while a few made it to the top list. This evaluation was made on the basis of security best practices. A few of these best practices are illustrated, based on EFFs report in Table 1 [10].

Included in this criteria are anonymity and decentralization due to their importance in improving the privacy and security of individuals communicating online [29], these best practices are further discussed below:

1. *Encryption:* Communication should be encrypted, including all links in the communication path and communication should be encrypted end-to-end.
2. *Authentication:* There should be a way for a user to verify the identity of their correspondent even if the link is compromised.
3. *Forward Secrecy:* Past communications are secure even if the encryption keys are stolen. Keys generated for encryption should be random and cannot be reconstructed. Generated keys should be deleted, and no access is given to any users long-term keys.
4. *Open Source:* The source code of the application should be available for security review and testing.
5. *Documentation:* The cryptography used by the application should be well defined and provides a clear and detailed explanation of the crypto.
6. *Anonymity:* The identities of those communicating are not exposed including their physical location.
7. *Decentralization:* All messages sent and received are not routed through a central server, and no centralized infrastructure is required to communicate preventing a single point failure.

Table 1 EFF Secure Messaging Scorecard.

	EFF Criteria					Added Criteria	
Applications	Encryption	Authentication	Forward Secrecy	Open Source	Documentation	Anonymity	Decentralization
Pidgin+OTR	yes	yes	yes	yes	yes	no	no
Skype	yes	no	no	no	no	no	yes/no
TextSecure	yes	yes	yes	yes	yes	no	no
TorChat	no	no	no	yes	no	yes	yes
CryptoCat	yes	yes	yes	yes	yes	no	no
AnonChat	yes	yes	yes	yes	yes	yes	yes

From Table 1, almost all applications reviewed in EFFs scorecard do not include anonymity in their security model. Moreover, they do not entirely include decentralization. These two features are essential in improving privacy, security and trust in instant messaging applications [22]. In our research, a prototype *AnonChat* was developed to include all features required to provide a fully anonymous and secure instant messaging. Using the criterion's provided by EFF and adding anonymity and decentralization, we provide a reliable way to improve privacy and security in instant messaging applications.

To implement a solution for the scenarios above, *AnonChat*[1] provides end-to-end encryption of messages sent between clients with anonymity features where all messages are sent within the Tor network using hidden services. Traffic is sent between two rendezvous points as illustrated in Figure 7. The foundation of this methodology is based on Tors location hidden services, Off-the-Record Messaging (OTR) for encryption and the Socialist Millionaire Protocol for authentication. Currently, there are very few implementations of OTR with SMP over Tor location hidden services for instant messaging. Our aim is to implement an instant messaging application with the following properties:

1. Users cannot be personally identified by their contacts or address.
2. Contact lists, message history or metadata cannot be accessed by an intruder or server.
3. All communication is encrypted, and previous messages cannot be decrypted even if the encryption key is known.
4. The system is completely decentralized, and no centralized server infrastructure is required.
5. Enable users to resist censorship or monitoring in an unfavorable network environment.
6. Registration of users is free, and no personal details are required.

[1] *AnonChat* source available at: https://www.github.com/tbenjis/AnonChat

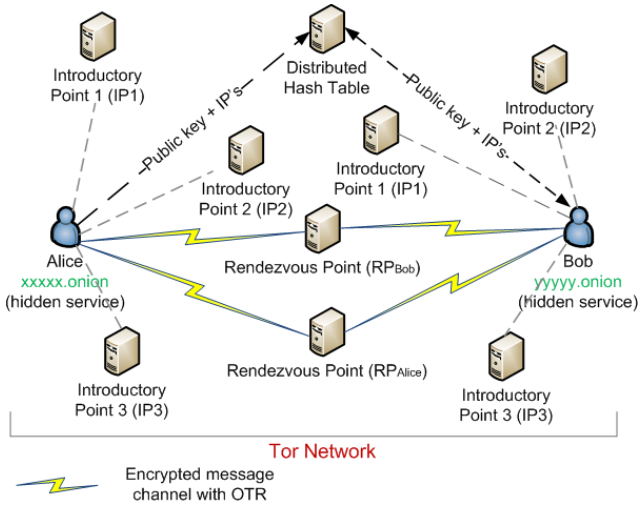

Fig. 7 Solution Approach

7. Users communicating can authenticate each other without exposing personal information.
8. Available on different platforms without the need for configuration by the user.

Providing security for users in communication by protecting their identities can be achieved by including anonymity into the messaging system. Security can also be improved by making the system completely decentralized. Our design goal is to prevent anyone from knowing whom Alice or Bob is by providing anonymity and decentralization. Also, to fully encrypt and implement forward secrecy for messages and provide authentication, so both parties are assured of whom they are communicating with. In this system, the two clients act as two different hidden service and client providing access to their servers on both ends. This allows the messages to be routed to two different rendezvous points in Tor. This provides improved security. Off-The-Record Messaging also provides encryption of these messages. The security protocol below describes how clients can be authenticated securely using Tor and in Off-The-Record messaging.

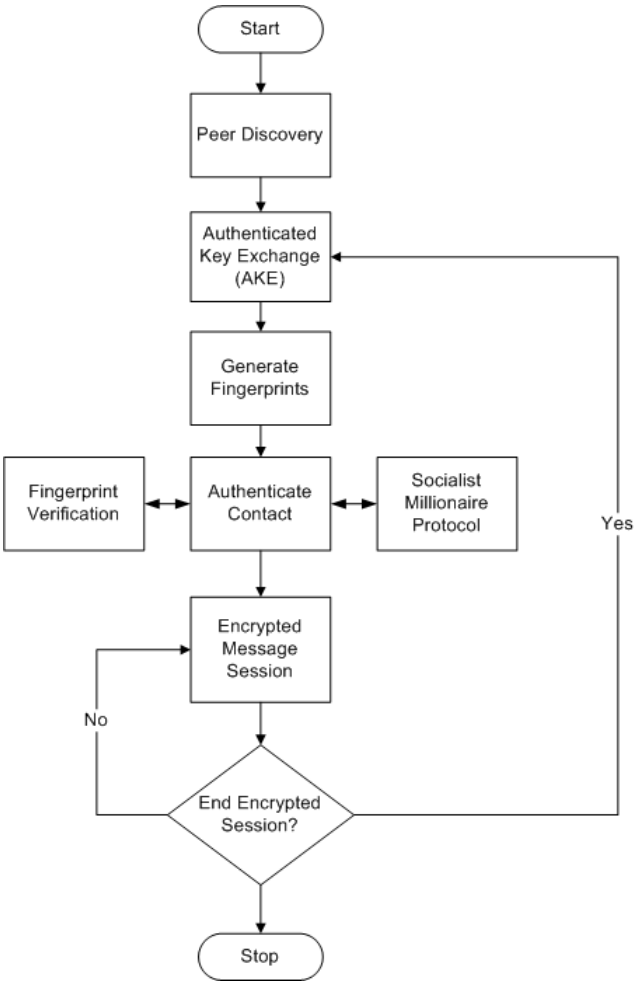

Fig. 8 Security mechanisms flow in *AnonChat*

3.3 Security Mechanisms

AnonChat provides several security mechanisms such as peer discovery, client authentication and encryption of messages. Here, the security mechanism used by the application will be discussed. Figure 8 depicts security mechanisms used in *AnonChat*. To initiate a conversation, clients need to authenticate first. *AnonChat* uses the fingerprint verification and Socialist Millionaire Protocol for authentication. Tor hidden service uses the hash of the public key in order to identify a client and to authenticate with the public key of that client. Security mechanisms are described below.

3.3.1 Peer Discovery

When a contact has been added, *AnonChat* sends a ping request with an authentication cookie to the onion address of the contact and then the contact responds back with a pong request for the authentication cookie. The authentication cookie is a random string of alphanumeric characters of 77 bytes generated by *AnonChat*. This length provides high entropy that is difficult to crack in months as discussed in [26]. As described in Figure 9 this process only allows contacts sending the correctly generated authentication cookie connect to the client. Figure 10 shows an example of an authentication cookie sent and received from *AnonChat* clients using a ping-pong request.

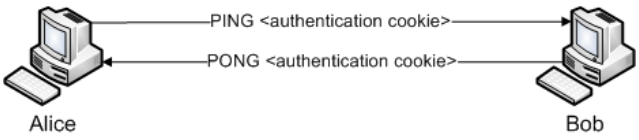

Fig. 9 Peer Discovery

```
ping oh47wpzo4dvnedup
0OQOXZT03TNXIDCKE0HGPFM00WN14N0E9DUP6LL0SB79D7
M15IN02QY89ZPZDU0U28QVVUBHP1CDM

pong
0OQOXZT03TNXIDCKE0HGPFM00WN14N0E9DUP6LL0SB79D7
M15IN02QY89ZPZDU0U28QVVUBHP1CDM
```

Fig. 10 A ping and pong request

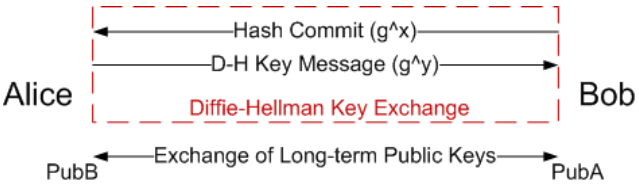

Fig. 11 AKE Process

3.3.2 Authenticated Key Exchange (AKE)

This process uses OTR to perform AKE. The idea is to allow two *AnonChat* clients perform a Diffie-Hellman (D-H) key exchange in order to create an encrypted channel and then perform a mutual authentication inside the channel [9]. This is the first stage of providing an encrypted communication via OTR in *AnonChat*, and it generates the long-term public keys used for fingerprint verification. Figure 11 describes the AKE process. To initialize AKE *AnonChat* client initiate an OTR conversation by sending the */otr* command and an OTR query message to initiate an encrypted communication. As described in the OTR protocol [1], to initiate the AKE, Alice and Bob pick random x, y respectively which is at least 320 bits. All exponentiations are done modulo a particular 1536-bit prime, and g is a generator of that group. Alice then signs g^x by using $AES_r(g^x), HASH(g^x)$ where r is a random value of 128 bits and the hash of g^x is computed [1] and sends

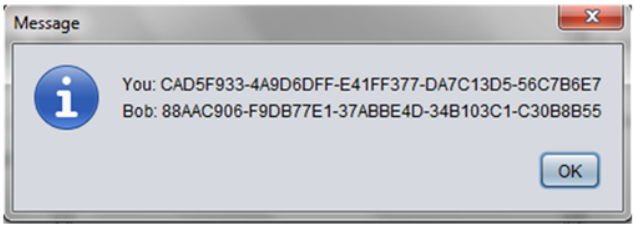

Fig. 12 Fingerprint Verification

to Bob. Next Bob also verifies that Alice's g^y is a legal value ($2 \leftarrow g^y \leftarrow modulus-2$) then computes and send Alice $r, AES_c(X_B), MAC_{m2}(AES_c(X_B))$ where X_B is $X_B = pub_B, keyid_B, sig_B(M_B)$ and pub_B representing the long term authentication public key of Bob and the MAC is a function computed on a message using a secret key shared between Alice and Bob [5]. This is represented in (Equation 1)

$$B \rightarrow A : r, AES_c(X_B), MAC_{m2}(AES_c(X_B)) \tag{1}$$

r is used to decrypt the value of g^x sent earlier and Alice then verifies id the value of $HASH(g^x)$ matches the one sent earlier. This process is described in OTR protocol version 3 [7]. Alice then computes M_A such that $M_A = MAC_{m1\prime}\ (g^y, g^x, pub_A, keyid_A)$ and $X_A = pub_A, keyed_A, sig_A(M_A)$ Alice then sends these values computed $AES_c\prime(X_A), MAC_{m2\prime}(AES_c\prime(X_A))$ to Bob as illustrated in (Equation 2) where the two AES keys c, c' and four MAC keys $m1, m1', m2, m2'$ are computed by hashing $s = (g^x)^y$ in various ways.

$$A \rightarrow B : AES_c\prime(X_A), MAC_{m2\prime}(AES_c\prime(X_A)) \tag{2}$$

Next Bob uses pub_A to verify $sig_A(M_A)$ if all the verification succeeds, they both generate $s = g^x y$, a shared secret and thus can verify that s is known by someone with the key pub_B which is Bob. Signatures authenticate the shared secret, not content. Here, the fingerprint for both client is generated and can be retrieved for verification.

3.3.3 Fingerprint Verification

AnonChat uses OTR fingerprint verification to authenticate contacts. Alice and Bob can verify their fingerprint by clicking the Show Fingerprint menu after an encrypted chat has been initiated. This is a one-way verification method and can be used to detect man-in-the-middle attacks. Fingerprint verification should be done via a different communication channel. Figure 12 shows a simple fingerprint verification in the prototype.

3.3.4 Implementing Socialist Millionaire Protocol (SMP)

This section discusses the implantation of the Socialist Millionaire Protocol. The goal of the Socialist Millionaire Protocol is to allow Alice and Bob compares a shared secret without disclosing the shared secret itself or any other information apart from whether the shared secret is equal or not. To setup an SMP process, all computations are done in a group G of large prime order q. The Diffie-Hellman group 5 which is a 1536-bit modulus group is used for this computation. The primary goal of the protocol is to determine if $x = y$ where x is a known secret of Alice and y is Bobs secret where both are elements of the set $\mathbb{Z}_q$ [1]. Generators g_2 and g_3 are created through Diffie-Hellman key exchange. Then Alice chooses $a_2 \ni_R \mathbb{Z}_q$ and Bob chooses $b_2 \ni_R \mathbb{Z}_q$ then they exchange $g_1^{a_2}$ and $g_1^{b_2}$ and they both compute $g_2 = g_1^{a_2 b_2}$. This process is repeated to generate g_3.

To bind x and y, Alice picks $a_2 \ni_R \mathbb{Z}_q$ and computes $(P_a, Q_a) = (g_3^a, g_1^a, g_2^x)$ then Bob computes P_b, Q_b respectively and they both exchange P_a, Q_a, P_b and Q_b. Finally, to determine whether $x = y$, the value a_3 of Alice is used to compute $R_a = \left(\frac{Q_a}{Q_b}\right)^{a_3}$. Then Bob computes $R_b = \left(\frac{Q_a}{Q_b}\right)^{b_3}$ and they both exchange R_a and R_b. Now, both Alice and Bob can now compute $R_{ab} = R_a^{b_3} = R_b^{a_3}$. Both parties can now verify that:

$$\begin{aligned} R_{ab} &= \left(\frac{Q_a}{Q_b}\right)^{a_3 b_3} \\ &= \left(g_1^{a-b} g_2^{x-y}\right)^{a_3 b_3} \\ &= g_3^{a-b} g_2^{(x-y) a_3 b_3} \\ &= \left(\frac{P_a}{P_b}\right) \left(g_2^{a_3 b_3}\right)^{x-y} \end{aligned} \tag{3}$$

To verify if $x = y$, they both need to check if $R_{ab} = \left(\frac{P_a}{P_b}\right)$ from (Equation 3). Since $(g_2^{a_3 b_3})$ is a random generator of G if $x = y$, then $\left(R_{ab}.\frac{P_a}{P_b}\right)$ will be equal to 1 [1]. This eventually leaves the result $1.(g_2^{a_3 b_3})$ which would be the same for both Alice and Bob, verifying that the secret both used are the same.

The Socialist Millionaire Protocol provides another form of authentication aside from the fingerprint verification. This makes authentication easier rather than manually comparing fingerprint every time. SMP allows two users to agree on a shared secret and then verify themselves with this shared secret without a third-party interfering or knowing what that shared secret is. *AnonChat* detects incoming SMP commands and initiates the required SMP process. Figure 13 shows how

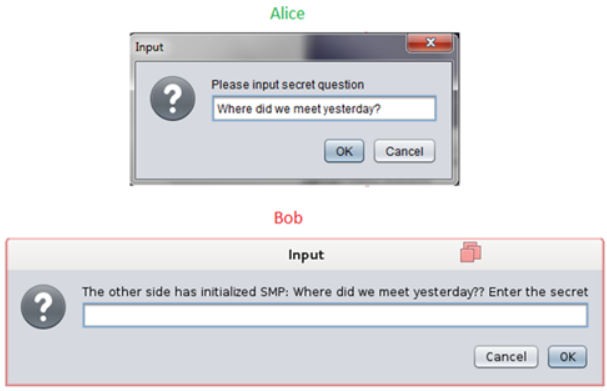

Fig. 13 SMP initiation and response dialog

a Socialist Millionaire Protocol request is initiated and responded to in *AnonChat*.

3.3.5 Message Encryption

After a successful AKE, message sent would be transmitted encrypted. The advantage of using OTR to encrypt messages sent in *AnonChat* is that OTR uses perfect forward secrecy where every message sent is encrypted using a completely new encryption key. This allows each message produce a unique ciphertext. The encryption process is shown below as described in OTR version 3 protocol [5]. All exponentiations are done, modulo a particular 1536-bit prime, The 1536-bit prime number, p, used for modulo operations in OTR is defined in RFC 3526 [31] under Group ID 2 and g is a generator of that group. Supposing Alice wants to send a message to Bob, the following process is taken to securely transmit the message and provide perfect forward secrecy as described in OTR protocol version 3 [7]:

Alice:

1. Alice picks her most recent Diffie-Hellman encryption keys that have been acknowledged by Bob. Here key_A represents the key, and $keyid_A$ is its serial number.
2. If key_A is Alice's most recent key, she generates a new D-H key ($next_{dh}$), to get the serial number $keyid_A + 1$.
3. Next Alice picks the most recent D-H key for Bob. key_B representing the key, and $keyid_B$ representing its serial number.
4. Alice then uses Diffie-Hellman to compute a shared secret from the two keys key_A and key_B, and generates the sending AES key, ek, and the MAC key, mk.

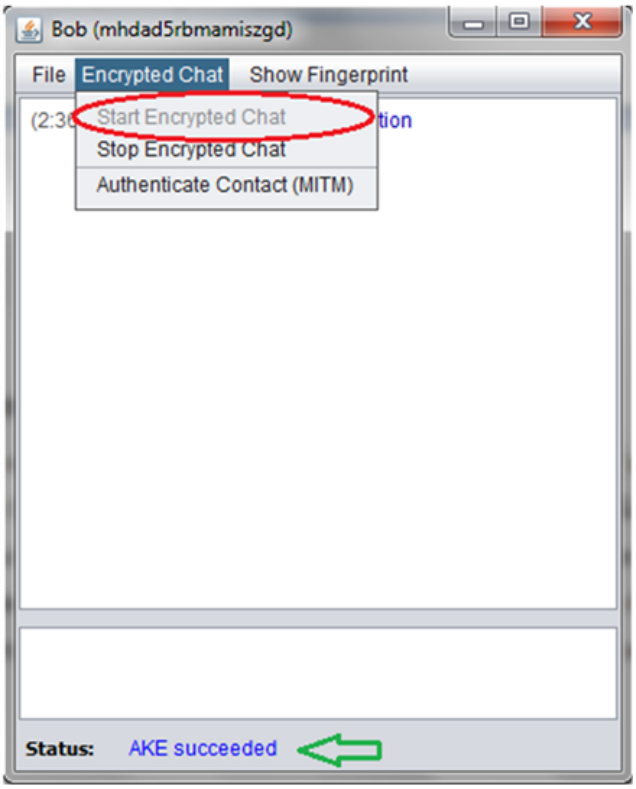

Fig. 14 An encrypted chat session

5. Old MAC keys *oldmackeys* are collected and never used again.
6. Alice then picks a value of the counter, ctr, so that the triple (key_A, key_B, ctr) is never the same for more than one Data Message Alice sends to Bob.
7. Alice computes T_A such that $T_A = (keyid, keyid_B, next_{dh}, ctr, AES - CTR_{ek,ctr}(msg))$ where msg is the message is being encrypted.
8. Alice then sends $T_A, MAC_{mk}(T_A), oldmackeys$ to Bob

Bob:

1. Bob then uses Diffie-Hellman to compute a shared secret from $keyid_A$ and $keyid_B$, and generates the receiving AES key, ek, and the receiving MAC key, mk.
2. Bob then uses mk to verify $MAC_{mk}(T_A)$.
3. Then uses ek and ctr to decrypt $AES - CTR_{ek,ctr}(msg)$.

/otr is sent with every message requiring encryption after the AKE has succeeded. Figure 15 shows a sample plain and encrypted text sent over the network using OTR. Message encryption is initiated using the Start Encrypted Chat menu on the chat window (See Figure 14). The application performs an Authenticated Key Exchange first before an encrypted session can begin.

```
/otr hello, how are you?

?OTR:AAIDAAAAAAEAAAACAAAAwJsbYqOD5ilzCz+bEF7
b0CZrdoRiGbviXGKZEXs6jgyezwsVqV+gsGSF1SSocJR
rfayIN4dyOyvgiwTjY2npNCKirpAhi0NJhkr6LqIPLHS
HsxP5s923RVCzzcmSUgs5JJXWPteHD4G9LxNeYBguSea
xK0tF7UZpo28RAQD7PlWt6ZGbQC0kSUZkcNP0+JRIgjS
oWaWMWhVLSgCyFdH9EIQo4Oo+QJqNkkGgvRVJmjlZ+ex
gQLi/GRRhxHUE0n1NKQAAAAAAAAACAAAAEvY5Ueh3XfZ
tJ0RqvfJSOyoeWmHs4y7H/uu2nCu6rj/3dsB8yAVCAAA
AAA==.
```

Fig. 15 A sample message encryption in *AnonChat*

4 Performance Measurement

In this section, the performance of *AnonChat* with TorChat will be compared. This is because both *AnonChat* and TorChat use Tor network as underlying protocol and other messaging applications use a different design in their underlying protocol. TorChat uses peer discovery to identify contacts before a message can be sent to them; this is similar with *AnonChat*. Although they both use same underlying protocol, since TorChat does not provide any form of encryption, an Authentication Key Exchange (AKE) test cannot be performed with TorChat.

To measure the performance of the application, several performance tests (20 each) at different geographic locations are performed to understand the effect of a location on the performance of both TorChat and *AnonChat*. This test was done using Digital Ocean cloud server, and a droplet of 1GB RAM, 30GB SSD, 1 Core CPU and 2-gigabit network interface was created. This droplet was relocated to four geographical regions; two droplets in America, one in Europe, and one in Asia, (New York, San Francisco, London, and Singapore). Other tests were performed using several network environments in Nicosia, North Cyprus on a Windows 64 bits machine with 4GB RAM, 2.10 GHz dual-core CPU and Linux 32 bits machine with 4GB RAM and a 2.10 GHz dual core CPU.

In Nicosia, North Cyprus, several network environments were simulated the SoftPerfect Connection Emulator [27]. The connection emulator imitates networks with low bandwidth, delays, and other network problems. With this packet, transversal issues can be simulated in order to mimic a low-grade communication channel. For each network environment, 20 simulated tests were performed, and the averages recorded as described later in this chapter. The five different network environment used to perform the simulated tests from the Windows machine are:

1. 56.0 Kbps modem
2. 128.0 Kbps Dual channel ISDN
3. 256.0 Kbps DSL, Frame Relay
4. 1.5 Mbps T1, DSL cable modem
5. 2 Mbps DSL Cable modem

In New York, San Francisco, London and Singapore, similar network environments and hardware were used to test the prototype (1GB RAM, 30GB SSD, 1 Core CPU and 512 MB swap space with a 2 gigabyte transfer rate on an Ubuntu 14.04 x64 operating system). The switching nodes on these servers are connected via a 2-gigabit interface connected to a switch and with a 10-gigabit network. To evaluate the performance of the application, analysis were done to determine the time it takes for the application to discover and connect to a contact in both *AnonChat* and TorChat and the time it take to perform an Authenticated Key Exchange in *AnonChat*.

4.1 Peer Discovery Test

In the peer discovery test, an analysis is performed in order to determine the time it takes for a contact to be detected online in TorChat and *AnonChat* right after the application has been initialized. Since TorChat uses similar protocol for identifying peers in the network, a comparison can be made on the performance of both applications. Averages calculated are based on five different network environments. Table 2 and Figure 16 shows the average time it takes to discover a contact online in these network environments. The average is calculated using (Equation 4).

$$P_t = \frac{1}{n} \sum_{i=1}^{n=20} p_i \tag{4}$$

P_t represents the average peer discovery time in seconds for each network environment. p_i is the peer discovery time for each trial in a particular network environment, and n represents the number of times this test was performed.

From Table 2, peer discovery takes approximately 50 seconds on a 56 Kbps network environment in *AnonChat* and about 60 seconds in TorChat. This time is further reduced in a 2 Mbps DSL cable modem network to approximately 40 seconds. The process of discovering peers involves locating the hidden service address and authenticating the contact in which a ping / pong request is sent during discovery. A few data items sent during this process include the online status, client ID, personal message, client version, profile name and other information.

Table 2 Average Peer discovery time and Standard Deviation per network

Networks (Cyprus)	Average (Pt) in Seconds (AnonChat)	Average (Pt) in Seconds (TorChat)	Standard Deviation (AnonChat)	Standard Deviation (TorChat)
56.0 Kbps	50.96	60.14	2.48	3.34
128.0 Kbps	46.07	45.51	3.10	3.46
256.0 Kbps	42.08	42.42	6.93	4.79
1.5 Mbps	41.55	42.02	4.93	3.42
2 Mbps	40.50	41.93	4.97	4.36

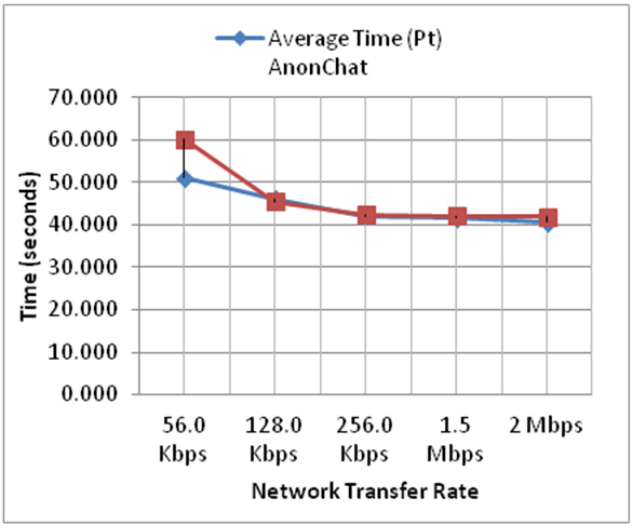

Fig. 16 Average peer discovery time per network

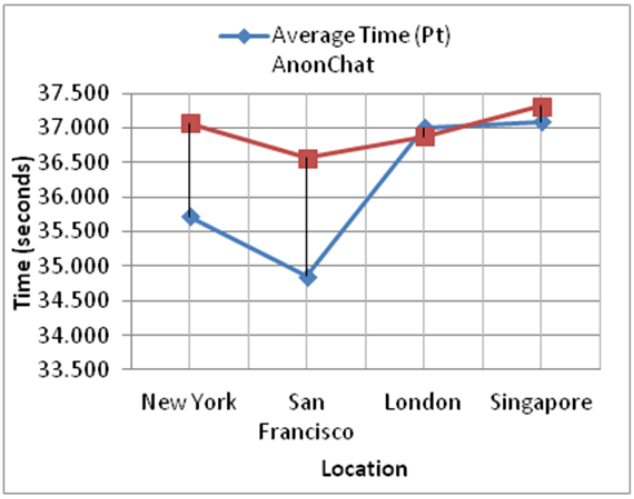

Fig. 17 Average Peer discovery time in various geo locations

In order to calculate the time it takes to discover peers in different geographic locations, tests were performed on similar Linux machines located at various geographical locations.

Table 3 and Figure 17 shows the average time required to discover online contacts from these locations. From Table 3 and Figure 17, it is observed that San Francisco is faster than others in discovering online contacts while Singapore and London take longer than East and West coast of the United States. This behavior is similar both in *AnonChat* and TorChat. In *AnonChat*, it takes approximately 35 seconds in San Francisco and New York, and it takes about 37 seconds for the same cities in TorChat. The latency is reduced in *AnonChat* by approximately 1.5 seconds.

The latency observed in both applications may be due to the delay in Tor relays selected in these locations. In this test, all measurements were taken from Nicosia, Cyprus to all other locations.

4.2 Authenticated Key Exchange Process Test

The AKE process involves several aspects from generating fingerprints to initiating an encrypted chat session using OTR. It involves both peers performing a Diffie-Hellman key exchange and exchange of long-term public keys securely used for authentication and not encryption. To test the AKE performance of *AnonChat*, a measurement is taken to determine the time it takes for both contacts to initiate an encrypted chat session in different network environments. This process involves the key exchange and fingerprint generation. TorChat does not contain authentication, so determining averages only on *AnonChat* is done. Averages are calculated using (Equation 5) as described below.

$$A_t = \frac{1}{n} \sum_{i=1}^{n=20} a_i \tag{5}$$

A_t represents the average authentication key exchange time in seconds for each network environment. a_i is the authentication key exchange time for each trial in a particular network environment, and n represents the number of times this test was performed.

In Table 4, it is observed that it takes a few seconds to initiate an encrypted chat session for the first time; this time is reduced as the network speed increases. In order to determine the average time required to initiate an AKE in *AnonChat*, the test was performed 20 times in which every trial is a new AKE process. (See Table 4

Table 3 Average Peer discovery time and Standard Deviation in various locations

Locations	Average (Pt) in Seconds (AnonChat)	Average (Pt) in Seconds (TorChat)	Standard Deviation (AnonChat)	Standard Deviation (TorChat)
New York	35.72	37.07	1.30	1.22
San Francisco	34.86	36.56	2.27	2.07
London	37.01	36.88	1.05	2.06
Singapore	37.09	37.31	1.31	2.30

Table 4 Average AKE process time and Standard Deviation on various networks

Networks (Cyprus)	Average (At) in Seconds	Standard Deviation (SD)
56.0 Kbps	4.89	1.22
128.0 Kbps	4.61	1.69
256.0 Kbps	4.52	1.00
1.5 Mbps	3.92	0.62
2 Mbps	3.86	0.59

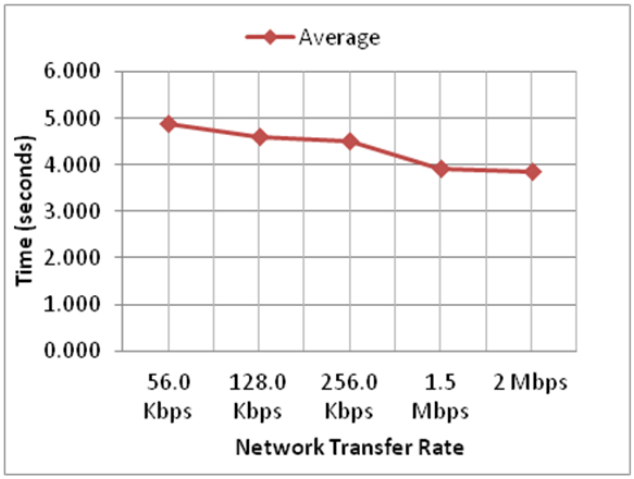

Fig. 18 Average AKE process graph

and Figure 18). From Figure 18, it can be seen that the time required to initiate an encrypted chat session decreases as bandwidth increases to approximately 4 seconds at higher bandwidths. The delay in lower bandwidths can be due to packet loss and network issues associated with low bandwidth networks.

Table 5 Average AKE process time and Standard Deviation at various locations

Locations	Average (At) in Seconds	Standard Deviation (SD)
New York	2.996	0.577
San Francisco	3.011	0.533
London	3.040	0.418
Singapore	3.036	0.430

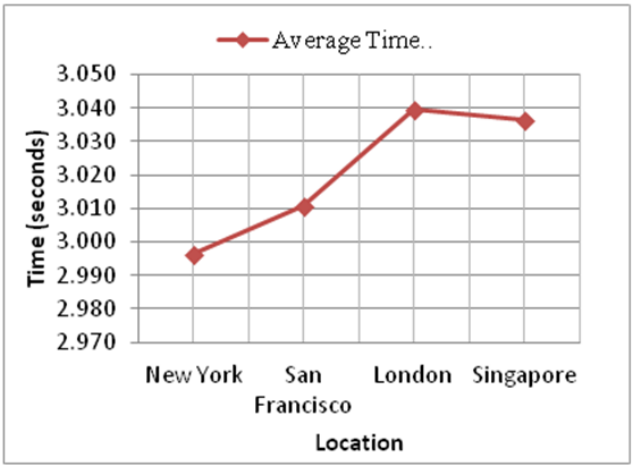

Fig. 19 Average AKE process time at various geo locations

Table 5 and Figure 19 shows that the Average time required to perform an AKE process in different locations is approximately 3 seconds. The fluctuations in the AKE process time are due to the latency in Tor relays as the packet travels through the network. Also, because Tor uses TCP, packet reconstruction can also cause a delay in some situations. From Figure 19, New York shows faster AKE process with about 2.9 seconds followed by San Francisco. These measurements are taken from Nicosia Cyprus to all other locations. These processing times are approximate values and can vary depending on network conditions. They all have the approximate time of 3 seconds for an AKE process.

4.3 AnonChat and TorChat Performance Comparison

From the performance tests done in Section 4.1 and 4.2, it is evaluated that the total time spent (T_M) from initialization to when a message is sent in *AnonChat* is given as (Equation 6) and in TorChat as (Equation 7) where T_{PD} is the peer discovery time, T_{AKE} is the AKE time, M_T is the time required to deliver a message in which M_T varies depending on message size and

network conditions, E_T is the time it takes to encrypt a message and n is the number of messages being sent.

$$T_M = T_{PD} + T_{AKE} + n(M_T + E_T) \tag{6}$$

$$T_M = T_{PD} + n(M_T) \tag{7}$$

Comparing the performance of *AnonChat* and TorChat, from (Equation 6) and (Equation 7) respectively, the time T_M of *AnonChat* can be calculated the averages from the performance tests using (Equation 8) where P_{t_i} is each peer discovery time in different locations as tabulated in Table 3.

$$T_{PD} = \frac{1}{n}\sum_{i=0}^{n} P_{t_i} \tag{8}$$

$$T_{AKE} = \frac{1}{n}\sum_{i=0}^{n} A_{t_i} \tag{9}$$

With (Equation 9) where A_{t_i} is each AKE time in different locations as shown in Table 5. Given these equations, T_{PD} can be evaluated as $T_{PD} = 36.17$ and $T_{AKE} = 3.02$, resulting in (Equation 10).

$$\begin{aligned} T_M &= 36.17 + 3.02 + n(M_T + E_T) \\ &= 39.19 + n(M_T + E_T) \end{aligned} \tag{10}$$

Using (Equation 11), T_M can be calculated for TorChat where $T_{PD} = 36.96$, from initialization such that:

$$T_M = 36.96 + n(M_T) \tag{11}$$

From (Equation 10), E_T which is the time it takes to encrypt a message varies and is negligible since and average encryption time tested takes about 23 milliseconds for a 4096 bytes message size on a typical machine. Also, since the messages size is limited and less than 4096 in *AnonChat*, E_T is evaluated as 0.023 seconds. Using M_T as a constant on both applications since it depends on network transfer rate and conditions $M_T = 1$ and $n = 1$ for a single message. Thus from (Equation 10), T_M is evaluated as $40.213sec$.

For TorChat, T_M can be evaluated from (Equation 11) as approximately 38 seconds in TorChat and 40 seconds in *AnonChat* to send a message from initialization. Note that this initialization process is only performed once. Even though this time is reduced in TorChat by approximately 2 seconds, the security features provided in *AnonChat* such as message encryption and authentication outweighs the 5.9% additional latency observed in *AnonChat*. This appears as a result of the security process (AKE and message encryption) in *AnonChat*.

5 Conclusion

This research seeks to improve security and anonymity in instant messaging applications using Tor anonymity network and Off-The-Record messaging to circumvent censorship and enhance privacy and security in communication. Detailed analysis of current messaging applications, their architecture and security is provided; communication and security protocols were analyzed, and a review of their security issues and recommendations were performed. Theoretical ideas for developing a secure and anonymous instant messaging application were defined and a prototype to support our idea was implemented. This prototype *AnonChat* uses perfect forward secrecy provided by OTR to prevent previous messages from being decrypted even when the private key is stolen. It also uses Tor hidden services to provide an anonymous peer-to-peer, client-server architecture and the Socialist Millionaire Protocol for authentication and preventing impersonation.

The analysis of this prototype was developed and tested in order to determine its performance under several network conditions. Peer discovery and Authenticated Key Exchange times are analyzed in different geographical locations (New York, San Francisco, London, and Singapore). This result shows that the speed of delivery is determined by both the speed of connection and the speed of interconnected Tor relays located in different parts of the world. The performance of *AnonChat* is compared with TorChat because of the similarity in their architecture and from the results, it was observed that an additional 2 seconds is required for initialization in *AnonChat* compared to TorChat. Despite this observed delay, the security benefits outweigh the latency observed during initialization.

The fundamental contribution of this research can be summarized as first characterizing the security requirements for an anonymous, secure and decentralized communication system. Then, providing low latency anonymous and secure instant messaging using Tor hidden services and evaluating its performance and security on high and low bandwidth network infrastructures. Furthermore, encryption using perfect forward secrecy and decentralization in which no server infrastructure is required was introduced with anonymity and authentication to instant messaging. As it is well-known that no system is completely secure, the difficulty of accessing private information by an intruder can be increased by constant improvement in the security of a system.

5.1 Future Work

Though this research provides a usable proof-of-concept implementation of an anonymous, secure and decentralized instant messaging application, further research will enhance and extend our findings.

Encrypted File Transfer: Currently, the prototype does not support encrypted file transfer. Providing an encrypted file transfer that supports OTR will also improve its usability and further provide a means of securing and sending various forms of data anonymously.

Further Improvement in Detecting Impersonation: Impersonation can still be an issue if the shared secret is known by a third party, providing further methods for detecting impersonation will significantly improve its security.

Additional Performance Improvement: *AnonChat* uses Tor hidden services as its transport; low-speed Tor relays can reduce the performance of the application by increasing authentication and peer discovery time. Further improvement in Tors network can improve how relays are assigned and reduce congestion on some relays that will eventually improve the performance of the application.

References

1. Alexander C, Goldberg I, Improved user authentication in off-the-record messaging. Proceedings of the 2007 ACM workshop on Privacy in electronic society - WPES 07. ACM Press, New York, New York, USA, p 41 (2007)
2. AlSabah M, Bauer K, Goldberg I, Enhancing Tors performance using real-time traffic classification. CCS 12 Proceedings of the 2012 ACM conference on Computer and communications security. pp 7384 (2012)
3. Bauer K, Mccoy D, Sherr M, Grunwald D, Experimen-Tor: A Testbed for Safe and Realistic Tor Experimentation. CSET (2011)
4. Berson T, Skype security evaluation. ALR 111 (2005)
5. Borisov N, Goldberg I, Brewer E, Off-the-Record Communication, or, Why Not To Use PGP. Proceedings of the 2004 ACM workshop on Privacy in the electronic society (WPES 04) (2004) doi: 10.1145/1029179.1029200
6. Boudot F, Schoenmakers B, Traor J, A fair and efficient solution to the socialist millionaires problem. Discrete Applied Mathematics 111:2336 (2001) doi: 10.1016/S0166-218X(00)00342-5
7. Cypherpunks.ca: Off-the-Record Messaging Protocol version 3. `https://otr.cypherpunks.ca/Protocol-v3-4.0.0.html` (2014). Accessed 21 Dec 2014
8. Dingledine R, Mathewson N, Syverson P: Tor: The second-generation onion router. SSYM04 Proceedings of the 13th conference on USENIX Security Symposium 13:21 (2004) doi: 10.1.1.4.6896
9. Di-Raimondo M, Gennaro R, Krawczyk H: Secure off-the-record messaging. In: Proceedings of the 2005 ACM workshop on Privacy in the electronic society - WPES 05. ACM Press, New York, New York, USA, p 81 (2005)
10. EFF: Secure Messaging Scorecard — Electronic Frontier Foundation. `https://www.eff.org/secure-messaging-scorecard` (2014). Accessed 22 Dec 2014
11. Egger C, Schlumberger J, Kruegel C, Vigna G: Practical attacks against the I2P network. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp 432451 (2013)
12. Erkkonen H, Erkkonen H, Larsson J, Larsson J: Anonymous Networks. EDA390 Computer Communication and Distributed Systems (2010)
13. Fernandez P: Through the Looking Glass: Envisioning New Library Technologies Securing Your Digital Library with Encryption. Library Hi Tech News 32: (2015)
14. Frosch T, Mainka C, J FB, Bader C, Horst G: How Secure is TextSecure? eprint.iacr.org (2014)
15. Geti2p.net: I2P Anonymous Network. `https://geti2p.net/en/docs/how/tech-intro` (2014). Accessed 20 Dec 2014
16. Goldberg I: On the security of the Tor authentication protocol. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp 316331 (2006)
17. Hindocha BN, Chien E: Malicious threats and vulnerabilities in instant messaging. Virus Bulletin Conference, vb2003 (2003)
18. Juan Pablo Timpanaro, Isabelle Chrisment OF: A Birds Eye View on the I2P Anonymous File-Sharing Environment. 6th International Conference, NSS 2012, Wuyishan, Fujian, China, November 21-23, 2012 Proceedings 7645:135148 (2012) doi: 10.1007/978-3-642-34601-9_11
19. Kobeissi N, Breault A: Cryptocat: Adopting Accessibility and Ease of Use as Security Properties. arXiv preprint arXiv:1306.5156 (2013)
20. Kurek K: Instant Messaging and Cross Site Scripting (XSS). GRIN (2012)
21. Loesing K: Tor Hidden Service Performance Improvements. Free Haven Project 19. (2009)
22. Motahari S, Ziavras SG, Jones Q: Online Anonymity Protection in Computer-Mediated Communication. IEEE Transactions on Information Forensics and Security 5:570580 (2010) doi: 10.1109/TIFS.2010.2051261
23. Overlier L, Syverson P: Locating hidden servers. In: 2006 IEEE Symposium on Security and Privacy (S&P06). IEEE, p 15 pp.114 (2006)
24. R. Mahy, P. Matthews JR: Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). Internet Engineering Task Force (IETF) (2010)
25. Rusitschka S, Gerdes C, Eger K: A low-cost alternative to smart metering infrastructure based on peer-to-peer technologies. In: 2009 6th International Conference on the European Energy Market. IEEE, pp 16 (2009)
26. Salem O, Hossain a., Kamala M: Intelligent system to measure the strength of authentication. 2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications, ICTTA (2008) doi: 10.1109/ICTTA.2008.4530020
27. SoftPerfect: SoftPerfect WAN Connection Emulator for Windows. `https://www.softperfect.com/products/connectionemulator/` (2014) Accessed 22 Dec 2014

28. Stedman R, Yoshida K, Goldberg I: A user study of off-the-record messaging. the 4th Symposium on Usable Privacy and Security 95104 (2008) doi: 10.1145/1408664.1408678
29. Swanson CM, Stinson DR: Extended results on privacy against coalitions of users in user-private information retrieval protocols. Cryptography and Communications (2015) doi: 10.1007/s12095-015-0125-x
30. Szilgyi P: Decentralized bootstrapping in clouds. In: 2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics, SISY 2012. pp 277281 (2012)
31. T. Kivinen and M. Kojo: More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE). Internet Engineering Task Force (IETF) Request for Comments 3526, May 2003 (2003)