Aatmanirbhar Sanchar: Self-Sufficient Communications

Jay Jhaveri, Abhay Gupta, Prem Chhabria, Neeraj Ochani, Sharmila Sengupta, Mrs. Sunita Suralkar, and Shashi Dugad

Abstract In the light of recent war crimes and data piracy conspiracies, privacy is of utmost importance to an organization and even to an individual. The majority of the population is dependent on third-party services for their daily communication. Albeit these major corporations advertise "secure" means of chat transfer, they install various kinds of backdoors to sell the user's data to advertisers. Under the notion of going "Aatmanirbhar" i.e., Make in India, we have developed an indie solution without incorporating any third-party services or APIs. "Aatmanirbhar Sanchar" aims at providing users with a real-time off-the-grid, secure, and anonymous messaging service. It features an End-to-End encrypted transmission of messages and data files likewise. This is achieved by combining the open-source AES algorithm with a self-developed XOR encryption process.

Keywords Messaging · Security · Privacy · Self-sufficient · Aatmanirbhar · Self-hosting · AES · Hash-based message authentication code verification

J. Jhaveri (✉) · A. Gupta · P. Chhabria · N. Ochani · S. Sengupta · Mrs. S. Suralkar
Computer Engineering, Vivekanand Education Society's Institute of Technology, Mumbai, India
e-mail: 2018.jay.jhaveri@ves.ac.in

A. Gupta
e-mail: 2018.abhay.gupta@ves.ac.in

P. Chhabria
e-mail: 2018.prem.chhabria@ves.ac.in

N. Ochani
e-mail: 2018.neeraj.ochani@ves.ac.in

S. Sengupta
e-mail: sharmila.sengupta@ves.ac.in

Mrs. S. Suralkar
e-mail: sunita.suralkar@ves.ac.in

S. Dugad
Tata Institute of Fundamental Research (TIFR), Mumbai, India
e-mail: shashi@tifr.res.in

J. Hemanth et al. (eds.), *Intelligent Cyber Physical Systems and Internet of Things*,
Engineering Cyber-Physical Systems and Critical Infrastructures 3,
https://doi.org/10.1007/978-3-031-18497-0_41

(HMAC) · Indigenous private server · Scalability · Cross plat-form · Throwaway · Anonymous

1 Introduction

In today's world, privacy and security are of utmost importance to an individual. Let me elaborate: Data Privacy and Data Theft are the hot debate in the World-Wide mass media at the moment, but have you ever wondered what exactly it is? Have you ever questioned how the so-called "Free" applications are kept afloat? They pay their bills by selling that very data you unknowingly give them while using their "free" services. This borderline stolen data is then used for targeted, personalized advertisements and much worse. To combat this, we are developing anonymous communication software without the use of any third-party services, hence maximizing the privacy of an individual. To put it simply, secure messaging is a way of safely exchanging documents between users, healthcare providers, organizations, and their customers.

2 Motivation

Currently, all other messaging services are hosted on Third-party cloud platforms, mainly Google and AWS cloud services. Let us consider WhatsApp for example, which has been recently acquired by The Facebook (META) team. After this acquisition, WhatsApp updated its privacy policy which gave access to Facebook to collect private information on its users causing many controversies and heated discussions in the IT industry around the globe.

There is a dire need for us to focus on these privacy problems faced by users using these "free" applications like WhatsApp and Facebook messenger. The companies owning these applications do not take adequate security measures in handling the user data but drive their marketing/advertising agenda through the data provided by their users.

Further, In the light of recent events, amidst the Russian-Ukraine War, there are major sanctions placed by the west on Russia, disabling them from using multiple western applications. Even their banking apps were restricted leading to a major downfall in their economy. Now, India has developed a native solution for the banking sector called the Unified Payments Interface, also known as UPI. Why not take this spirit and create an indie chat application?

These were mainly our inspiration to create India's very own messaging service without utilizing any kind of third-party services. In collaboration with the Tata Institute of Fundamental Research (TIFR), we have built a secure communication, cross-platform messaging application wherein a user can exchange vital information with other users and groups of users without being concerned about any kind of data leak or data monetization.

3 Literature Review

Cohn-Gordon et al. [1] in "A Formal Security Analysis of the Signal Messaging Protocol" explained that Signal Protocol is a private messaging protocol that provides instant messaging encryption to applications such as Skype, Facebook Messenger, and WhatsApp among many others, with more than 1 billion active users. The signal contains unique unfamiliar security features (such as "future privacy" or "post-compromise security"), which are made possible by ratcheting, a process through which session keys are updated with each new message.

Singh et al. [2] in "Blockchain-Enabled End-to-End Encryption for Instant Messaging Applications" presented a blockchain-based E2EE framework for mitigating current messaging application vulnerabilities. During the installation of the application, the end-user device generates a pair of public/private keys and asks its mobile network operator to issue a digital Identifier and store it in the blockchain. The end user can obtain another user's certificate from their chat private server and utilize a ratchet forward encryption process to interact securely with them.

Botha1 et al. [3] in "A Comparison of Chat Applications in Terms of Security and Privacy, ECCWS 2019 18th European Conference on Cyber Warfare and Security" described a gadget that helps people adapt to social life by allowing them to understand domain messages, names, letters used in mailboxes, in daily newspapers, and so on. The major goal of the project is to solve the above problem by using a Raspberry Pi and an OCR sensor to recognize environmental messages automatically and then using TTS to translate those messages into voice or audio for better and easier engagement with society.

Sabah et al. [4] in "Developing an End-to-End Secure Chat Application" presented a chat program that provides end-to-end security, allowing users to safely transmit confidential information without fear of data loss. In addition to the storage protection. This article presents a list of requirements for creating a secure chat application, and the program was created based on these requirements. The suggested chat application was compared to other popular apps based on those criteria, and it was also put to the test as a proof of concept for delivering End-to-End security.

Burak [5] in "Encryption Methods and Comparison of Popular Chat Applications" proposed end-to-end encryption chat solutions that allow users to safely transmit personal information. The paper includes a list of requirements for developing a secure chat application.

Canetti [6] in "Universally Composable End-to-End Secure Messaging" explained all the contemporary widely accepted encryption algorithms in detail and their limitations in the real practical world. It also helped in choosing the most suitable encryption algorithm for this chat application.

Emura [7] in "Membership Privacy for Asynchronous Group Messaging" focuses on a method capable of hiding membership information from the viewpoint of non-group members in a secure group messaging (SGM) protocol, which we call "membership privacy".

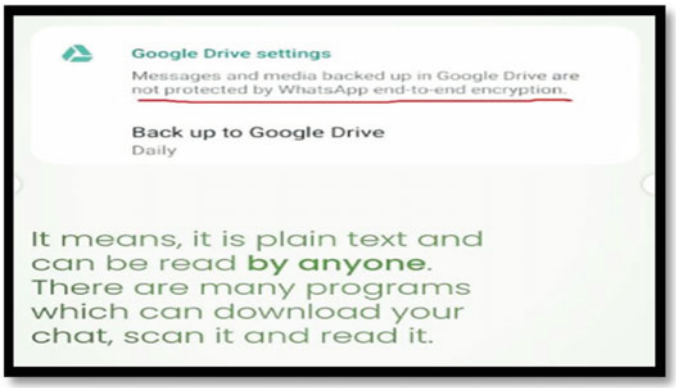

Fig. 1 WhatsApp Plain text Backup Proof

4 Lacuna in the Existing System

"Data is the new fuel" and major tech corporations are utilizing every gizmo at their disposal to amass and utilize user data for monetary advantage because their customers' personal and behavioral data is worth millions of dollars if mined to its full potential.

WhatsApp and other Applications [8] provide the option of verifying users' public keys, but the mechanisms used are not robust and pose major session hijacking issues. Besides this, there is no reliable third-party involvement to check the suitability of keys stored on WhatsApp servers [9–13].

The backup method utilized by WhatsApp does not provide real end-to-end encryption (see Fig. 1). The alternate copy is kept in plain text on the user's cloud, depending on the user's OS, such as iCloud, Google Drive, One Drive, and so on.

5 Methodology

5.1 Joining a Chat Room

When a user visits the web application at http://aatmanirbhar-sanchar.live/, hosted on the private servers at Vivekanand Education Society's Institute of Technology (VESIT), he is greeted with the homepage asking for a Username along with a Room key (RK) (see Fig. 2). The entered username will act as the main identity of that particular user for the ongoing session. The room key is the most significant aspect of the chat application. This key serves a dual purpose:

1. The hash of the key acts as the identity of a particular group chat created using the same.

Fig. 2 Home page

2. The key itself is used in the encoding and decoding process of incoming and outgoing messages.

When the user enters a room key (RK), the key first goes through an extensive algorithm to test the strength of the key. If the resulting strength is not up to the standards for secure communication, it will warn the user of a weak key and the user may decide if he wants to proceed or add a new key.

A passphrase is highly recommended instead of a password to ensure utmost privacy while communicating. To keep it user-friendly, a random passphrase generator has been added. Now, when the user clicks the join button, the following processes occur (see Fig. 3):

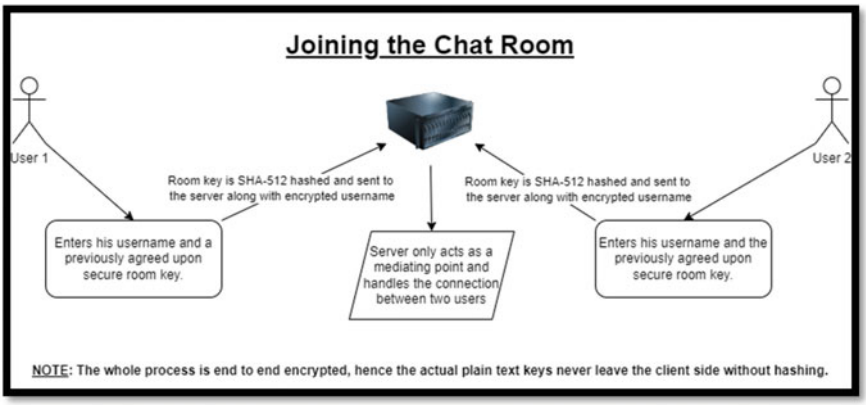

Fig. 3 Joining process

1. The key is first hashed using the SHA-256 algorithm to get ready to be transmitted to the server.
2. *NOTE:* The plain text key never leaves the client side
3. The username, AES encrypted using the RK, is appended along with the hashed RK and is sent to the server.

The user can now share this room key with the intended recipients to begin a secure private communication channel with him/her.

5.2 The Ephemeral Chat Room

When a user joins a chat room using the shared RK, the active user counter is incremented and their encrypted username is broadcasted to all the users active in the room utilizing which a greeting message is displayed (see Fig. 4).

Now every user is subscribed to the following events:

1. Join Response: *Handles a new incoming user.*
2. Chat Response: *Handles incoming text messages.*
3. File Response: *Handles incoming files.*
4. Leave Response: *Handles a user leaving.*

When a user joins a chat room using the shared RK, the active user counter is incremented and encrypted.

Fig. 4 The chat room

5.3 *Encryption Process*

Aatma Sanchar uses a double-layered encryption process for achieving enhanced security. The first layer constitutes the self-developed XOR encryption process:

This encryption system is based on the concept that if an object is XOR'ed by the same key twice it will revert to its original state. To make this viable in this innovative era of cybercriminals vs cybersecurity, multiple iterations of permutations and combinations on the original entity take place before further encoding. To put it simply, a text message is first converted to its binary format in the shape of matrices. These matrices are then shuffled and reshuffled to increase protection followed by undergoing the XOR process by the room key 10.

This encryption system is based on the concept that if an object is XOR'ed by the same key twice it will revert to its original state (Fig. 5). To make this viable in this innovative era of cybercriminals vs cybersecurity we have added multiple iterations of permutations and combinations of the original entity. To put it simply, a text message is first converted to its binary format in the shape of matrices. These matrices are then shuffled and reshuffled to increase protection followed by it getting XOR'd by the room key [9].

This encryption layer is followed by the Advanced Encryption Standard (AES-256) algorithm 11 to ensure privacy while maintaining efficiency. From the Graph in Fig. 6, it is clear that while there exist faster encryption algorithms other than AES, as the file size increases (which is a common situation in a messaging platform), AES easily comes out on top. Hence, AES-256 was selected as the second layer in this encryption process.

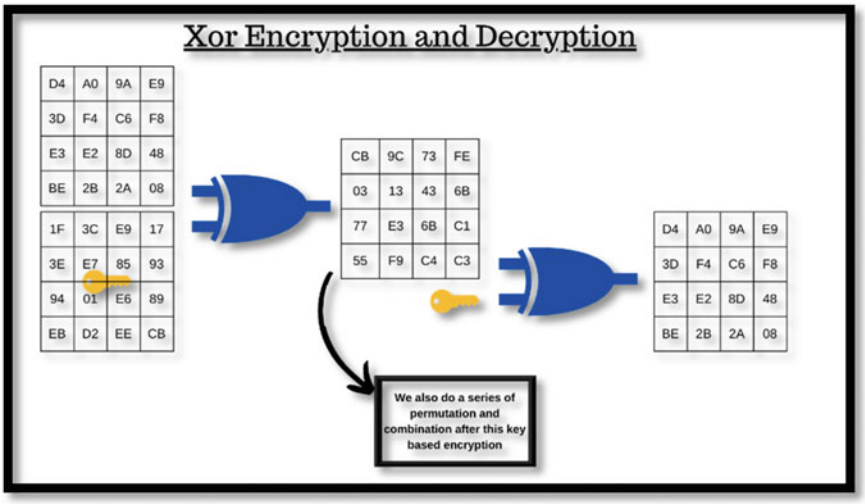

Fig. 5 Second layer encryption using the XOR matrix method

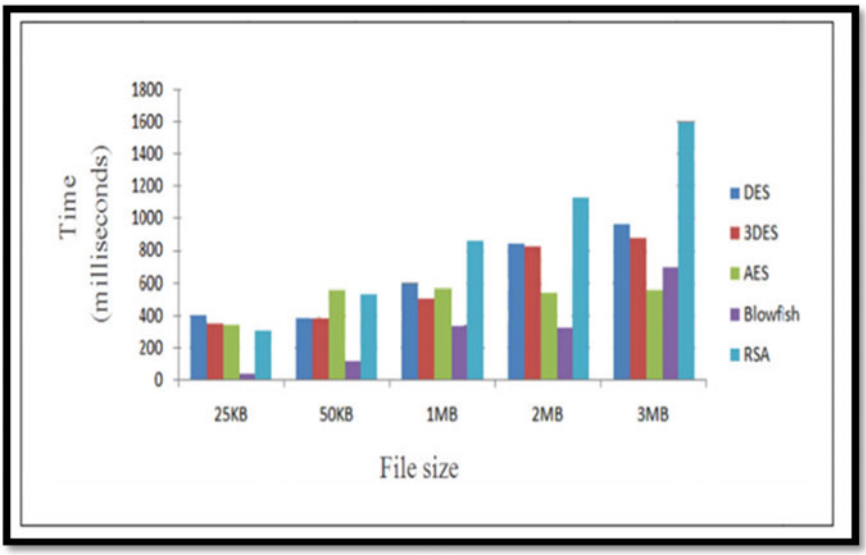

Fig. 6 Second layer encryption using the XOR matrix method

Finally, to ensure the integrity and authenticity of the transmitted message, the process of Hash-based Message Authentication Code verification (HMAC) is also practiced which guarantees tamper-proof messaging 12. This is done by creating a hash-based checksum using the combination of the room key and the data ready to be sent to the sender's client. This checksum value is then recomputed on the receiver's device, and if any discrepancy is detected it indicates that the message was tampered with (Fig. 13).

5.4 Transmitting a Message

The users can either directly type or send a text message using the provided chat box, else a user can also attach files up to the 50 MB limit to be transmitted.

1. Sending a text message (see Fig. 7):
   a. When a user types a message and hits the send button the message is encrypted using the Room Key (RK) utilizing the doubly layered encryption algorithm mentioned before. Further, the SHA-256 hashed RK and the encrypted Username are appended into a dictionary along with the encoded message.
   b. Next, we use the HMAC (hash-based message authentication code) algorithm to ensure the authenticity and integrity of the message being sent. The HMAC is generated using the above-created dictionary and the Room Key (RK).

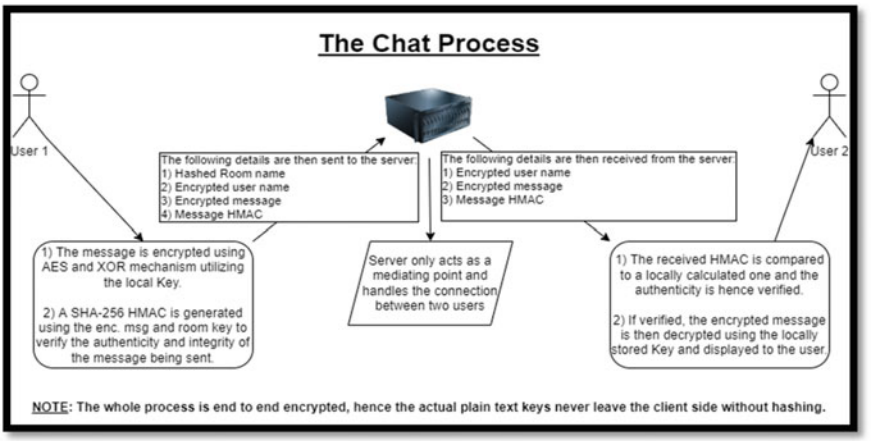

Fig. 7 Sending a text message

c. Finally, this encoded dictionary along with the HMAC appended to it is emitted through a socket to the server.

2. Sending a File as an Attachment (see Fig. 8):

a. A user also has the option to send any type of file as long as it is under the 50 MB limit. When he/she selects the file to be uploaded, the file is first converted into its binary (Base 64) format.

b. This binary format is encrypted using the Dual layer encryption process and is stored in the dictionary along with its file name and file type. Finally, as in the text messaging process, an HMAC code is calculated utilizing the above

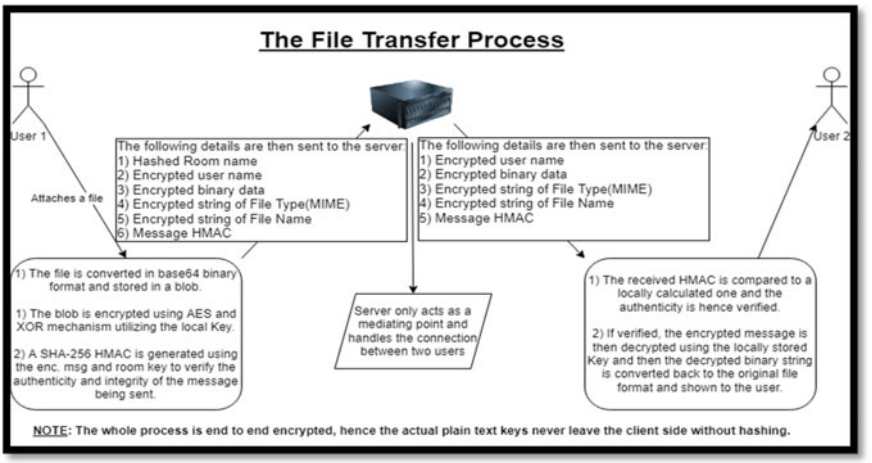

Fig. 8 File transfer process

dictionary and the room key. This is then sent using the same socket method as for a text message.

5.5 Receiving a Message

After a user has sent the encrypted message to the server then broadcasts the message to all the users currently connected to that particular room. On the receiver's end before any decryption process can start, the HMAC is recalculated on the client end utilizing the encrypted dictionary and if any disruption is found, an error is displayed in the chat box indicating the message was tampered with (Further explained in the Cryptanalysis part of the paper).

After verifying the HMAC code, the actual decryption process starts:

1. Receiving a text message (see Fig. 7):
   a. The incoming encoded dictionary is first decrypted using the AES algorithm followed by the reverse XOR method.
   b. This decrypted message is shown to the user in the chat box along with the decrypted username of the sender (see Fig. 4).
2. Receiving a file as an attachment (see Fig. 8):
   a. The incoming encoded dictionary is first decrypted using the AES algorithm followed by the reverse XOR method.
   b. This generates the file in its pure binary (Base 64) format. This binary file is then converted according to the MIME type into its original form.
   c. This original form is then converted into a blob link from where the receiver can download the same. If the File type is in a known multimedia format (Music, Image, Video), then the user is also given the option to preview the same within the chat box itself (see Fig. 9).

5.6 Leaving the Chat Room

A user can press the "leave" button to securely exit the chat room. Once pressed, the client is unsubscribed from all the live-time events and finally emits an encoded dictionary constituting the hashed room key and the encrypted user name. This username is then broadcasted to every active user with a message indicating that this person left the chat room.

Once left, the user cannot access the chat history and for enhanced security, the chats are never stored on the local device. If every user leaves a particular chat room, the session is destroyed in instantly.

Fig. 9 File preview in the chat room (*light mode*)

5.7 Experimental Environment Details

The following tools and technologies have been used in the development of Aatmanirbhar Sanchar:

- Front-End Development:

  React JS
  HTML/CSS

- Backend Development (server-side):

  Languages used:

  – NodeJS
  – Socket.IO

  Compatible Operating System:

  – Ubuntu (16.04)
  – Windows 10

  Requirements:

  – A Public Static IP to host the messaging application.
  – Android Studio for mobile applications (Auto File Sync).
  – Maps and Google Sheets API

6 Applications

6.1 In Large Organizations as a Quick, Secure Chat Platform

Our chat application being ephemeral does not store any of the chat data on the local client machine or the cloud server. This, apart from making our product more secure from any kind of unauthorized access to the local machines, also saves a lot of vital storage space in the cloud servers that can be essential for more important subjects. Being a throwaway chat application, it can be reused N number of times without any load on the server or the client.

6.2 Communications Where Security is of National Importance

In matters where national security is of utmost importance, one cannot simply rely on the external, untrustworthy third-party application for secure communication [11]. Our chat application has up-to-date encryption algorithms along with our self-developed XOR encryption process ensuring utmost protection without using any kind of third-party material. For instance, during a hostage situation, the officers in command can safely plan a rescue operation along with their subordinates without sacrificing the vital game plan to anyone intercepting the conversation.

6.3 **Aatmanirbhar Samakraman***: File Auto-Synchronization App*

Using our encryption process, we have also developed an auto synchronization file app useful in situations where the user repeatedly stores important readings in the format of a file in a selected directory and wants to securely upload the same to a remote server.

The user first selects a specific directory to be continuously monitored by the app. He/she then selects another directory where he wants these files to be moved once uploaded to the server. He then presses the start syncing button to activate the background process (see Fig. 10).

Now every time a new file is stored in that specific directory it triggers the application and the file are automatically uploaded to the remote servers and the locally stored file is moved to the second directory. This transfer of the file to the server takes place using the Multi-Part technology after being encrypted with our XOR-encryption process.

Fig. 10 Aatmanirbhar Samakraman android application

Along with the file, the current location coordinates of the user are also sent which is then used to display a tracking history on a web-based map UI for easy analysis: (http://file.aatmanirbhar-sanchar.live/) (see Fig. 11).

7 Results

In the end, we were successful in developing a secure, private, ephemeral chat application and deployed it on a private server hosted at our college, the VESIT campus. The web application is completely free from third-party services and is fully built upon open-source libraries.

We also developed an encryption system from scratch. This XOR encryption process is nothing but dividing the binary data into matrices followed by shuffling and reshuffling of the data and finally the data being XOR'd by a predefined key (see Fig. 5).

Finally, the output generated from the XOR method is passed to the Advanced Encryption Standard (AES-256) Algorithm in turn ensuring utmost security.

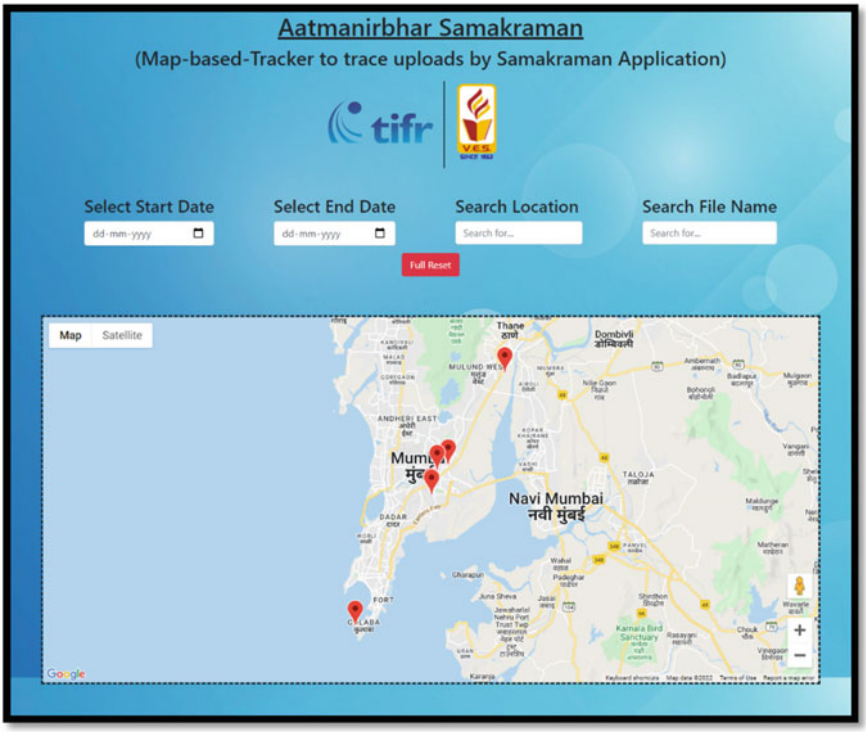

Fig. 11 Aatmanirbhar Samakraman live-map tracking

Finally, to ensure the integrity and authenticity of the transmitted message, the process of HMAC (hash-based message authentication code) verification is also practiced which guarantees tamper-proof messaging. To test the security of our product, we have tried the following Cryptanalysis techniques:

1. Snooping
   a. An intruder listens to traffic between two machines on a network in a snooping attack. We prevent this attack by only transmitting everything in an encrypted format. To an outsider, everything will look like gibberish (see Fig. 12).

2. Man in the middle attack (MITM)
   a. An attacker intercepts a message/key sent between two communicating parties through a secure channel in this sort of attack and tries to alter them. We can detect any tampering in the incoming messages due to the HMAC verification process. If any alterations are detected, a "decryption error" is shown to all the users indicating a tampered message (see Fig. 13).

3. Server attack
   a. In the case wherein an intruder has successfully gained access to the remote server would cause no harm to the privacy of the users. This is achieved due

Fig. 12 Intercepted message as visible to an intruder

Fig. 13 Error is shown indicating message tamper

to the fact that no vital information is stored or even transmitted to the server without it being dually encrypted. So, the worse this intruder can do is shut down the server itself.

4. TCP-SYN flood attack
   a. TCP-SYN flood is a type of DDOS attack where the attacker starts pinging the server continuously from several different IP addresses by not providing Full information which is required by the server. Due to this, the server has to disconnect the running applications and wait for the partially opened connection started by the mugger, which can take enough resources to render the system unusable to authorized congestion.

7.1 Comparison of Results with Existing Systems

One of the major differences between Aatmanirbhar Sanchar and other similar applications is the promise of keeping your data safe and keeping the application open source for anyone to verify its' contents. The majority of the existing chat applications keep their source code proprietary and hide data mining loopholes in their terms and conditions (Fig. 14.).

Fig. 14 Data leaks in existing software

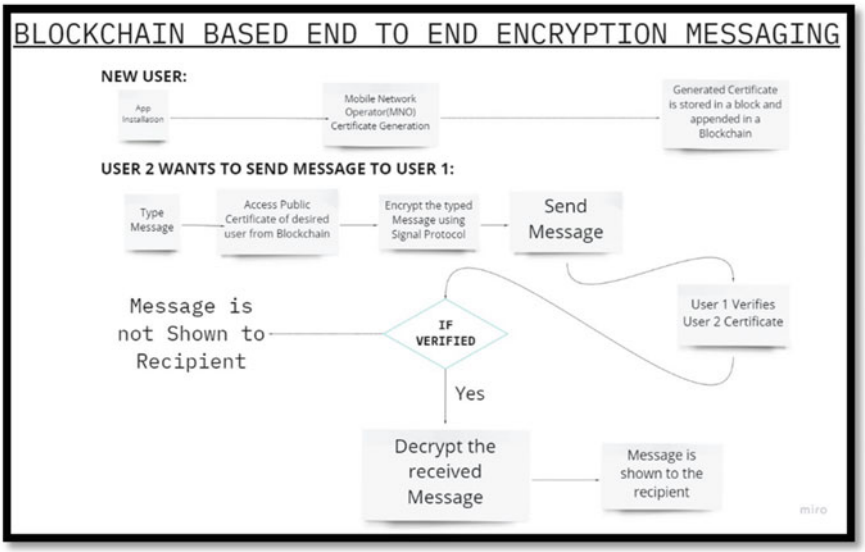

Fig. 15 Decentralized chat application

8 Future Scope

8.1 Converting the Application to a Decentralized WEB3 Application (See Fig. 15)

Now, instead of a centralized server, we can also theoretically use a blockchain network to convert the application into a web3 decentralized application.

Blockchain is a type of database. In this database, data is stored in the form of blocks and these blocks are chained together to form a blockchain. When each block in the chain is added to the chain, it is given a precise timestamp.

It is extremely difficult to modify the contents of a block after it has been put into the blockchain. This is because each block has its own hash in addition to the previous block's hash.

So, in this, if a new user installs the app for the first time a certificate will be generated which is then stored in a block and appended to the blockchain. This certificate will be used as a public key to the blockchain system.

If a user wants to send a message to a particular person the certificate of the recipient is accessed from the blockchain. The message will be encrypted using the signal protocol before being sent. Now, user 1 verifies user 2's certificate, and once verified, user 2 will be allowed to decrypt the message otherwise an error should be thrown.

9 Conclusion

Secure and Private communication is a serious issue in today's world. One is not able to communicate with their loved ones without being spied upon by the "Mark Zuckerbergs" of the world. There is a serious lack of an indie Secure Communication application that can be freely and securely used by public and private organizations.

Hence, we are very excited to present to the world, "Aatmanirbhar Sanchar: An Ephemeral, Anonymous, Secure Chat Application" based on a custom-based encryption algorithm that can be easily hosted on one's very own private servers without any external eyes watching over.

Conflict of Interest Statement On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Cohn-Gordon K, Cremers C, Dowling B, Garratt L, Stebila D (2020) A formal security analysis of the signal messaging protocol. J Cryptol 33(4):1914–1983. https://doi.org/10.1007/s00145-020-09360-1
2. Singh R, Tewari H (2021) Blockchain-enabled end-to-end encryption for instant messaging applications. https://arxiv.org/abs/2104.08494
3. Botha JG, Van 't Wout MC, Leenen L (2019) A comparison of chat applications in terms of security and privacy. In: 18th European conference on cyber warfare and security. University of Coimbra, Portugal
4. Sabah N, Kadhim JM, Dhannoon BN (2017) Developing an end-to-end secure chat application. Int J Comput Sci Netw Secur
5. Burak M (2021) Encryption methods and comparison of popular chat applications. Adv Artif Intell Res 52–59
6. Emura K, Kajita K, Nojima R, Ogawa K, Ohtake G (2022) Membership privacy for asynchronous group messaging. National Institute of Information and Communications Technology (NICT), Japan.
7. Canetti R, Jain P, Swanberg M, Varia M (2022) Membership privacy for asynchronous group messaging. National Institute of Information and Communications Technology (NICT), Japan
8. Marlinspike M, Perrin T (2016) The X3DH key agreement protocol. Signal
9. Whatsapp whitepaper (2021) WhatsApp encryption overview. Whatsapp. https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf
10. Perrin T, Marlinspike M (2016) The double ratchet algorithm. Signal
11. Daemen J, Rijmen V (1999) AES proposal: Rijndael. Rijndael Block Cipher
12. Bellare M, Canetti R, Krawczyk H (1996) Message authentication using hash functions—the HMAC construction. RSA Lab CryptoBytes
13. Hess A (2015) Encryption and cyber security for mobile electronic communication devices. Fed Bur Inv