Preventing Data Leakage by Securing Chat Session with Randomized Session ID

Husna Sarirah Husin[1] and Fatin Nabila Muhamad Riduan[2]

[1]Universiti Kuala Lumpur, Malaysian Institute of Information Technology, Malaysia

Abstract— Messaging applications have become one of the largest and most popular smartphone applications. It includes the capacity for the users to communicate between themselves via text messages, photos and files. It is necessary to safeguard all messages. Privacy is one of the biggest issues which most individuals in the world of instant messaging ignore. Although several instant messaging applications offer varying security for users, the weaknesses and danger of data assault are increasing. they tend to fail. Not just business discussions, our data must also be safeguarded during everyday discussions since data is very sensitive for everybody, and data protection is very crucial to prevent undesired loss of information. To address these types of weaknesses and hazards associated with data attacks, we require an encrypted messaging protocol and also hide IP address method for a safe interaction. This paper's goal is to protect conversations from targeted attacker by securing the communication between user and hide IP address from hackers.

Keywords—data leakage. messaging application, secure session, session ID

1. Introduction

Nowadays, Malaysians prefer using online shopping platform or hire a rider to purchase the item for them. This is happening because of the pandemic Covid19 that cause Malaysians not to go out frequently. This situation makes the demand for the rider and courier drastically increased. Some of them have become exceptionally popular such as Foodpanda, Grab, Poslaju, Lalamove and Bungkusit. To deliver the item to the customer the rider or courier they need to know the phone number of the customer to contact. When customer receive the call, they do not know whether it is the real or fake person that deliver their item. It is because when the customer shares their phone number, they do not know to whom it is passed on.

Besides, the smartphone that we use today should uncomplicatedly connect us to each other. Mostly message application that exists provide simple solutions which are often free of charge. However, even the message application is free we still have to pay a price for it which is we cannot see. The price is our phone number and contacts not only that the metadata usually used commercially by the operators. When the information about when we communicate with whom and for how long, money can be made from this information. This increasing reliance on the mobile chat service as well as the surge of number of attacks and vulnerabilities are main concerns and have seen much demand for security solutions [1].

Furthermore, another concern for messaging application is the data of the users leaked accidently or by person who may have some bad intention. This data leaked may cause great loss to the users that are using that messaging application because from this data hacker can get more details information regarding that person such as phone number, email address, identity number and home address. For instance, when the hacker gets the phone number of the targeted users, they can easily spoof a legitimate business email account and asked the users to send sensitive information to them.

In addition to spoof a legitimate business data to users through phone number for data leakage, messaging application that provided phone number and email address are at high risk for data leakage. Since hacker get the phone number and email address, they can attack the users with phishing attack. It is the same technique as spoof they sent some link to users and users can simply clicking on the link given that contains malicious code that allow the hacker to access the users phone. In other words, the first step that attacker need is basic information from data leakage of message application that contain phone number and from there they can get more information they need.

This paper proposes a secure chat session and preventing data leakage using a random session ID. In the proposed system, users do not need to know the other users phone number or email address to communicate. Moreover, as a step to ensure privacy, the proposed system will automatically delete conversation within 24 hours. The objectives of this paper are:

- To design an application to secure chat session from data leakage
- To develop the random session ID within chat conversation between user to prevent data leakage
- To test whether the random session ID without phone number and email address for chat conversation secure from data leakage

However, this system has few limitations such as that it would need a stable Internet connection for it to be able to pull the recent data from server, and it can only support in Android operating system. In addition, the message reminder depends on a secure internet connection to be able to warn the user when it is prompted.

2. Related Work

2.1. Chat session

A session is essentially a conversation. A session takes place over a network, and it begins when two devices acknowledge each other and open a virtual connection. It ends when the two devices have obtained the information, they need from each other and send "finished" messages, terminating the connection, much like if two people are texting each other, and they close the conversation by saying, "Talk to you later." The connection can also time out due to inactivity, like if two people are texting and simply stop responding to each other.

A session can either be a set period of time, or it can last for as long as the two parties are communicating. If the former, the session will expire after a certain amount of time; in the context of TLS encryption, the two devices would then have to exchange information and generate new session keys to reopen the connection. Chat may be delivered through text, verbal, audio, visual or audio-visual (A/V) communication via

the Internet. If conducted through a desktop, chat requires software that supports Internet Relay Chat (IRC) or an instant messenger application, where a central server manages chat communication between different end user clients. There are also online chat services that require users to sign up with a valid email address. After signing up, a user may join a group chat room or send a private message to another individual. Online chat services have purpose-built chat interfaces that manage the entire communication processes Chat conversation of message come in different type of platforms. There are two commonly used types of chat. They are Social Networks and Messaging Application.

Social networks are a platform where we can network, share ideas and communicate with everyone from around the world. Social networks are a business proprietor platform used to advertise and sell their brands and products.

A social networking site is a social media site that allows you to connect with people who have similar interests and backgrounds. Facebook, Twitter, and Instagram are three of the most popular examples of a social network website. These platforms allow us to connect with our friends, family, and even brands. Most social network sites let users share thoughts, upload photos and videos, and participate in groups of interest.

For illustration, Facebook Massager falls on social networks type of chat session because Facebook Messenger is a function of instant messaging built into Facebook. Originally launched in 2011, Messenger 's success has given rise to a dedicated application and website which emerged in 2014. The app and website is an instant messaging service that connects with the Facebook database and replaces the Facebook messaging service in-app.

On the other hand, messaging apps are much more private than social media, it's more restrictive and requires more a person's personal knowledge before we can reach them.

A messaging application is a mobile-phone-based software program that allows users to send and receive information using their phone’s internet connection. Messaging apps can transmit or receive a much wider range of data types than Short Message Service (SMS) or Multimedia Messaging Service (MMS). In addition to voice calls, video calls and text, messaging-app users can send and receive files, images, audio, location data, emojis and (in some cases) documents. For SMS service, One Time Password (OTP) is one of the typical multi-factor authentication and authorization mechanism.

Messaging apps were primarily designed for private communication between individuals or small groups, but are increasingly being used in new ways, including:

- Broadcast or bulk messaging: The capacity to send messages or other content to a large number of people.
- Encryption: End-to-end encryption means that the content of a message can be viewed only by the people sending and receiving messages. It cannot be decrypted and read by the company itself.
- A chatbot, also known as smart bot is a computer program that provide responds using text or voice and understand human language [3].

For example, application that under messaging application are WhatsApp which are the most popular messaging application, Telegram, WeChat, Snapchat and Line.

2.2. Threats and Vulnerabilities in Chat Session

A threat refers to the hypothetical event where the vulnerability is being used by an attacker. Normally, the threat itself will involve an exploit, as it is a common way hacker will make their move. After assessing what will bring the most reward, a hacker may use multiple exploits at the same time. Meanwhile, in order to force software to act in ways it is not intended to, vulnerabilities can be leveraged, such as gathering information about the current security defenses in place.

Instant messaging is a threat to malware that is emerging as a carrier. More and more individuals, both for personal and business reasons, are using instant messaging. Networks for instant messaging provide the ability not only to transfer text messages, but also to transfer files. Consequently, worms and other malware can be transferred by instant messengers.

Privacy threats caused in the workplace by messaging application clients include leakage of personal data, exposure to IP addresses, loss of confidential information, and eavesdropping. The same personal information that users either do not mind or do not know about sharing with the world when they sign up at home for messaging application services is becoming a more serious workplace problem.

On the other hand, poor session management may be the cause for broken authentication that would allow attackers to compromise session tokens [4] that would be the cause of phishing attack.

A series of researches reveals that unfortunately many social media applications, including chat applications, breach privacy of the users and are prone to different type of vulnerabilities for their users. Vulnerabilities include revealing plain-text passwords or storage of private information on the servers which can be revealed to non-authenticated users [5].

Data leakage is about an attacker manipulating or forging messages and sender information but not hijacking the entire account. In these messages are created and sent with a fake (spoofed) sender ID and bypasses user-identification mechanisms inside the application [6].

Many instant messaging protocols use various servers for user authentication and real chat message transmission. Such designs enable an instant messenger client to authenticate to a specific server, retrieve a session cookie and then use this session cookie to log in to secondary servers, such as the messaging server. However, this session cookie can be used to impersonate the authenticated user if it is captured via network sniffing.

One can hijack an instant messaging connection or impersonate other users in a variety of ways. Some methods are based on the inherently insecure nature of TCP/IP and related protocols, while others are specific to the design of the instant messaging protocol. These techniques necessitate that secure instant messaging clients utilize methods of data authentication to ensure that the data truly originates from the supposed source [7].

2.3. Current Technology to solve the Chat Session vulnerabilities

Encryption is the scrambling of plaintext messages, turning it into unreadable code that can only be deciphered by those who have the secret key. End-to-End Encryption is one of the most commonly used technologies to secure and send information across the internet. Hardware embedded into phones and computers allows for the random locks and keys that make

E2EE only work on the devices involved in the conversation [8].

An original message that is sent by sender is known as the Plaintext. The text which is coded is known as the Ciphertext. Encryption is the process of converting the plaintext to ciphertext. Decryption is the process of obtaining the plain text from cipher text. The cryptographic system works in two ways. In public key cryptosystem, two keys are used where private key stays with the sender and the public key is accessible to all the members in the network. In symmetric key cryptography, the data is sent from sender to receiver and same key is used for both encryption and decryption [9].

Signal protocol is a cryptographic protocol use to provide encryption for text, audio, video calls and different group messaging conversation. It is designed by Open Whisper Systems, and is the basis of Whatsapp End – to – End Encryption. This is a basic protocol which is used to provide end to end encryption. Signal Protocol's goals include end-to-end encryption as well as advanced security properties such as perfect forward secrecy and "future secrecy". The Signal cryptographic protocol has seen high take-up of encryption in close to personal communication through messaging services [9].

According to researcher the basic algorithm used in Signal Protocol is Curve-25519 and ECDH Protocol. Curve25519 is a cryptographic algorithm which offer 128-bit encryption and mainly use for ECDH key arrangement. Its security depends on trouble of discrete logarithm issue in large finite groups. Curve25519 was explicitly structured with the goal that secure, quick usage is simpler to create. Specifically, no validation of public keys is required and point multiplication can be proficiently processed utilizing a constant number of tasks which avoids a number of side-channel attacks [9].

Elliptic curve Diffie-Hellman is a key managing protocol allows different parties having elliptic key pair to share a secret key over a non-reliable medium. This shared secret may directly use as a key by receiver or can be used to derive another key. It characterizes how keys ought to be produced and exchanged between at least two parties. It accepts input as two keys and give yield in structure secret key [9].

2.4. Data leakage in Chat Session

In particular, mobile messaging apps have become one of the most common communication solutions in business and personal life, but have also been in the news regularly due to leakage of personal information and security concerns. Most recently, there are reports of possible backdoor in WhatsApp as well as security issues with Telegram, with some reports even claiming that the app has been compromised by the FSB. As messaging apps can become an unmonitored mechanism for a malicious insider to leak data via a variety of methods, this is potentially the biggest risk. The user could copy email text or capture file attachment screenshots, and infiltrate them securely via the messaging solution. The concern here is again that we face solutions providing end-to-end encryption.

End-to-end encryption is a communication method where the messages sent and received can be read by only the communicating parties. The messages cannot be read by third parties such as hackers, governments, internet providers, other individuals on the same Wi-Fi, or companies that provide the messaging service because only the communicating parties have the cryptographic keys required to decipher the messages.

3. Development of The Chat Session

Chat Session is a mobile application. Android Studio is the platform on which this app is built. As we all know, Android Studio is the official integrated programming environment (IDE) for developing Android apps. A Gradle-based build framework, debugger, code models, and GitHub integration are all included in Android Studio. Android Studio is a software application that runs on Windows, Linux, Mac, and other platforms. This program would use Firebase as its database. Google developed Firebase as a framework for developing smartphone apps. Both Cloud Firestore and Real-time Database can be included in this program. Cloud Firestore is the newest database for mobile app creation, while Firebase's original database is Real-time Database

3.1. System Architecture

The system architecture of the system is described as follows:

i. Creating or composing message requests. For making notification requests, the Notifications composer provides a GUI-based alternative. We must create message requests in a trusted server environment that supports the Firebase Admin SDK or the FCM server protocols for complete automation and support for all message types.

ii. The FCM backend accepts message requests, performs message fanout through topics, and creates message metadata such as the message ID, among other things.

iii. A transport layer at the network level that routes the message to the intended system, manages message execution, and implements platform-specific configuration as required. This transport layer contains the following components:

- Android transport layer (ATL) for Google Play services on Android devices
- APNs are Apple's push notification service for iOS smartphones.
- For mobile applications, the web push protocol is used.

The warning is shown or the message is handled by the FCM SDK on the user's screen, depending on the app's foreground/background state and any related application logic.

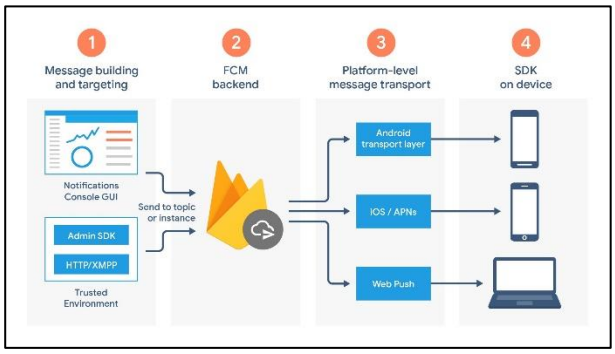

Figure 1. System Design of Application in Firebase

3.2. Activity Diagram

The modules in the proposed system are explain in Table I.

Table 1: Modules in the system

Module	*Description*
Session	Chat Session start when user click on button create session ID, if not successful user needs to click again on button to create session. When user successful create session ID encryptions (Extended Triple Diffie-Hellman) on session ID will happen and user can start copy session ID to share with other user, enter name,

Module	*Description*
	click on notification to receive notify when new message incoming. After that user will directed to page chatting where user can start chatting with share or receive session ID.
Messaage	Once user start a new session, this application will be creating a path to each session which is called as The Onion Router (TOR) path. This path will hide user IP address where the network will go through three hop networks before it reaches receiver destination. If the message successful, message will be sent and user can start to send text, media and documents. User can also view the list chat, list of media and also can change their name. Otherwise, message not successful user will get error message that indicate to user message cannot be send to receiver.
Recovery phase	User can recover their chat with recovery phase. Before user clear all chat user need to copy recovery phase that has been given and save it. Then user can click on button recovery phase and paste recovery phase that has been save. If the key match with the previous session, decryption of the key happened and chat will be recovering. Recovery error will appear when the key not match with previous session then the recovery not successful.
Disappearing message	When user start chatting, they can decide when the message will be deleted automatically. User need to go to chat setting and choose disappearing message and they can start choose when the message will be deleted whether 5 seconds, 10 second and etc. The message will be deleted on both side the sender and receiver.

3.3. Database

The following table 2 describes the tables that were created for this application.

Table 2: Tables in the database system

Table	*Description*
Job	Job table has one primary key which is jobid and two attributes create time is when the session ID created and serialized data are the process of turning data into strings
User	User table has serverID as primary key and display name and public key is the entities of the UseDatabase. These attributes will be called back as user insert the data.
Recipient	Recipient table has recipientID as primary key. In this table there are four attributes which are notification where user get notify if there are new incoming message, next entities expire message where user can setup when the message will be deleted from the chat. Other than that, registration and display name also included in this table.
Attachment	This table has many to one relationship with MMS table. There are almost 10 attributes inside this table because mostly these attributes for media. When user sent media for instance image, the data of the image like size, file _name, width, height, directory and caption will be recorded.
Thread	This database has many to one relationship with Message table. These table contain date of the message, message_id, status of the message whether successfully sent or not, when message expires, error message, type of the message, how many that message did not read by receiver and also count the message that has been sent and received
Backup	This database has one to one relationship with Job table and Message table. When user want recover the chat user need to have jobID to recover chat that has attributes of file size and timestamp of the message.

4. Security Testing

Over the past few years, mobile technology has evolved at an unprecedented rate, with a huge increase in the number of users. Mobile apps store and process a wide range of sensitive data, including credit card documents, intellectual property, and medical records. With the distinctions between protected and exposed data becoming increasingly blurred, a more robust and agile security architecture is needed.

Mobile Application Security Testing tests an application's security, as well as a large range of mobile application threat vectors, in order to detect potential vulnerabilities to ensure that the app is safe when in operation. In order to do the security testing for Chat Session application, we used tools that called Wireshark and SonarQube scanner to scan the vulnerability, security hotspot bugs and code smell in the source code of application. The next part will describe the security testing performed, the issues to be overcome and the recommendation to solve the issue.

4.1. Vulnerabilities Details

Issue - Use a dynamically-generated, random IV
Risk – Critical
Description - When using Cipher Block Chaining (CBC) for encryption, the Initialization Vector (IV) must be volatile and unknown. Otherwise, crypto-analysis attacks like the "Chosen-Plaintext Attack" will compromise the encrypted meaning. Since the IV's function is to ensure that the same plaintext encrypted twice yields two separate ciphertexts, it should be correlated with only one encryption period.
Recommendation – IVs should be:

- Volatile
- Random
- it's publishable (IVs are frequently published)
- A Message Authentication Code was used to authenticate the ciphertext as well as the ciphertext (MAC)

4.2. Bugs Details

4.2.1. Double Brace Initialization should not be used

Risk – Minor
Description - DBI generates an anonymous class with a reference to the owning object's instance; however, if the anonymous inner class is returned and retained by other objects, it will cause memory leaks. Even if there isn't a breach, DBI is so cryptic that most maintainers would be perplexed.
Recommendation – Instead, use Arrays.asList for lists, or manually connect each object to the collection.

4.2.2. Conditionally executed code should be reachable

Risk – Major
Description – Conditional expressions which are always true or false can lead to dead code. Such code is always buggy and should never be used in production. This rule would not cause any problems in any of the following scenarios:
Where the condition is a single boolean value at the end
When the condition is either true or false in the literal sense
Recommendation - When the condition is a single final Boolean or when the condition is simply true or incorrect, this law would not cause any problems.

4.2.3. Parameter 1 to this call is marked "androidx.annotation.NonNull" but null could be passed

Description – @NotNull, @NonNull, and @Nonnull fields, parameters, and return values are considered to have non-null values and are not normally null-checked before use. Setting one of these values to null, or failing to set one of these class fields in a constructor, can result in NullPointerExceptions at runtime.

Impact – Unless exception handling (on some platforms) is usable and introduced, NULL pointer dereferences normally result in the method failing. And where error management is used, returning the program to a stable state of service can be very difficult.
Recommendation – Double-check if all functions that return a value aren't empty before acting on the results.

4.3. Security Hotspot

4.3.1. Make sure this weak hash algorithm is not used in a sensitive context here (Weak Cryptography)

Risk – Medium
Description – Since collisions are likely, cryptographic hash algorithms such as MD2, MD4, MD5, MD6, HAVAL-128, HMAC-MD5, DSA (which uses SHA-1), RIPEMD, RIPEMD-128, RIPEMD-160, HMACRIPEMD160, and SHA-1 are no longer considered stable (little computational effort is enough to find two or more different inputs that produce the same hash).
Recommendation - Safer alternatives, such as SHA-256, SHA-512, and SHA-3, are preferred, and for password hashing, algorithms that do not compute "quickly," such as bcrypt, scrypt, argon2, or pbkdf2, are also better because brute force attacks are slowed.

4.3.2. Make sure the regex used here, which is vulnerable to polynomial runtime due to backtracking, cannot lead to denial of service (Denial of Service (DoS))

Risk – Medium
Description – When processing an input, most regular expression engines use backtracking to try all possible execution paths of the regular expression. This can trigger performance problems, which are known as disastrous backtracking scenarios. In the worst-case scenario, the regular expression's complexity grows exponentially with the scale of the data, which ensures that a small carefully-crafted input (such as 20 characters) will induce disastrous backtracking and device denial of service. With, in this case, a broad carefully-crafted input, super-linear regex complexity will have the same effect (thousands chars).
Recommendation – The maximum number of predicted repetitions should be defined using the appropriate quantifier, such as 1,5 instead of +.
Refactor nested quantifiers to minimize the number of ways the inner group can be matched by the outer quantifier. For example, in this nested quantifier case (ba+)+, the inner group can only be matched if there is exactly one b char each repeat of the group.

4.3.3. Make sure this debug feature is deactivated before delivering the code in production (Insecure Configuration)

Risk – Low
Description – Debug functions of a program make it easier for developers to detect glitches, which helps attackers as well. It also provides access to comprehensive information about both the application's running system and its customers.
Recommendation - Debug functionality cannot be activated on production servers.

4.4. Code Smell

4.4.1. Class variable fields should not have public accessibility

Risk – Minor
Description – The encapsulation theory is violated by public level vector fields, which has three major drawbacks:
Additional functionality, such as validation, is not possible to add.
The internal representation is visible, and it can't be modified later.
Member values will shift at any point in the code and does not match the programmer's expectations
Recommendation - Unauthorized changes are avoided from using private attributes and accessor methods (set and get).

4.4.2. Unused "private" fields should be removed

Risk – Major
Description – A private field that is declared but never included in the application is called dead code and should be deleted. This will boost maintainability because developers will no longer be confused about the purpose of the variable.
This law does not allow for reflection, which ensures that problems would be addressed on private fields that can only be reached through the reflection API.
Recommendation - The Java serialization runtime assigns a serialVersionUID to each serializable class, which is used during deserialization to ensure that the sender and recipient of a serialized object have loaded classes for that object that are serialization compatible.
A serializable class will declare its own serialVersionUID directly by declaring a static, final, and long field called serialVersionUID.

4.4.3. Inheritance tree of classes should not be too deep

Risk – Major
Description – One of the most important principles of object-oriented programming is inheritance. It's a method of compartmentalizing and reusing code by generating classes, which are arrays of attributes and behaviors that can be built on previously generated classes. However, violating this principle by constructing a deep inheritance tree will result in source code that is both complicated and unmaintainable. The most common cause of a too-deep inheritance tree is poor object-oriented architecture, which leads to a deliberate use of 'inheritance' where 'composition' would be more appropriate.
A Message Authentication Code was used to authenticate the ciphertext as well as the ciphertext (MAC).

5. Conclusion

The proposed system is able to create random session ID for users to ensure that the session is secure. This function is very useful for riders who deliver parcels. For instance, users buy item online and rider need to deliver the item both parties just need to share their session id with QR code only.
Other than that, without using phone number or email address the device IP address never exposed to the other users that we are communicating with and also the server that holding the data. This can ensure the information of the users is protected and secure with minimize metadata leakage.
Besides, this application also will automatically delete conversation within 24 hours. It is because the item that rider or courier hold to deliver to the customer is within 24 hours and there are many parcels every day that they need to deliver. With this application it can make riders works easier because after the parcels deliver the conversion between the customer with automatically deleted. The riders do not need to manual delete conversation as the new days come, they need to contact the new customer.

Future enhancement for this proposed system is to have it available in other language besides English. Other functions that can be added include GPS feature that allows users to share their location with other users, as well as function for users to record and send the voice recording to others users.

References

[1] H.-C. Chen, C.-H. Mao, Y.-T. L. Lin, T.-L. Kung, and C.-E. Weng, "A secure group-based mobile chat protocol," *J. Ambient Intell. Humaniz. Comput.*, vol. 7, no. 1, pp. 693–703, 2016.

[2] A. R. L. Reyes, E. D. Festijo, and R. P. Medina, "Securing One Time Password (OTP) for MultiFactor Out-of-Band Authentication through a 128-bit Blowfish Algorithm," *Int. J. Commun. Networks Inf. Secur.*, vol. 10, no. 1, 2018.

[3] A. Khanna, B. Pandey, K. Vashishta, K. Kalia, B. Pradeepkumar, and T. Das, "A Study of Today's A.I. through Chatbots and Rediscovery of Machine Intelligence," *Int. J. u-and e-Service, Sci. Technol.*, vol. 8, no. 7, pp. 277–284, 2015.

[4] K. Thakur, S. Juan, and A.-S. K. Pathan, "Innovations of Phishing Defense: The Mechanism, Measurement and Defense Strategies," *Int. J. Commun. Networks Inf. Secur.*, vol. 10, no. 1, 2018.

[5] P. Dashtinejad, "Security System for Mobile Messaging Applications," KTH University, 2015.

[6] R. Gupta and S. D. Joshi, "Common Vulnerabilities and Risk Analysis in Messaging Applications in Social Media with special reference to Whatsapp," *Int. J. New Innov. Eng. Technol.*, vol. 5, no. 3, pp. 32–37, 2016.

[7] N. Hindocha and E. Chien, "Malicious Threats and Vulnerabilities in Instant Messaging," California, 2003.

[8] R. Endeley, "End-to-End Encryption in Messaging Services and National Security—Case of WhatsApp Messenger," *J. Inf. Secur.*, vol. 9, no. 1, 2018.

[9] M. Manasa, D. V. Reddy, A. Yaswanth, and G. V. . Raj Kumar, "Encryption Techniques for Different Messenger Applications," *Int. J. Innov. Technol. Explor. Eng.*, vol. 9, no. 23, pp. 295–297, 2019.

.