Thesis

- [ ] **What is E2E Encryption.**

End-to-end encryption is a security measure used to protect the privacy of communication over the internet by encrypting the messages in such a way that only the sender and the recipient can read them. In end-to-end encryption, the messages are encrypted at the sender's end, and only the recipient can decrypt them at the other end. This means that even the service provider, network administrators, or anyone else who might intercept the messages can't read the message contents.

In messaging applications, end-to-end encryption is used to ensure the privacy and security of messages sent between users. End-to-end encryption can help to prevent eavesdropping, data breaches, and other forms of unauthorised access.

In a traditional messaging system, the service provider has access to the message content because the messages are stored in an unencrypted form at some point during their journey. Most non-E2E messaging applications use TLS to encrypt the messages between the server and the client device. That ensures the security of the message over the internet, but not the privacy of the sender and recipient, as the service provider will also have access to the conversation.

The main difference between traditional messaging systems and end-to-end encrypted systems is that the former allows service providers to access the message content, while the latter encrypts the message content in such a way that only the sender and the recipient can access it. This makes end-to-end encrypted systems much more secure and private than traditional messaging systems.

Some messaging applications like Signal not only provide end-to-end encryption of message content but also protect **metadata**. Metadata refers to the information about the message itself, such as who sent it, who received it, and when it was sent. This metadata can be valuable to attackers or third parties who wish to track users' communication patterns, build social graphs, or identify patterns of behavior.

For example, metadata can reveal information such as who a user communicates with frequently, the frequency of communication, and the timing of communication. This information can be used to build a social graph of the user and identify their social connections. Attackers or third parties can use this information to gain insights into users' social relationships and potentially use this information for malicious purposes such as targeted advertising or identity theft.

Metadata protection is important for protecting individual freedoms, such as the freedom of speech, by ensuring that individuals are able to communicate freely without fear of being tracked or monitored. This is particularly important for individuals in professions where privacy and confidentiality are crucial, such as journalists and doctors.

For example, journalists rely on the protection of sources to investigate and report on issues of public interest. If their communication patterns are monitored or their sources are revealed, they may face retaliation or persecution, which can inhibit their ability to report on important stories. Metadata protection can help prevent this by ensuring that communication patterns are not tracked and that sources remain anonymous.

Similarly, doctors rely on the protection of patient confidentiality to provide quality healthcare. If their communication patterns or patient information is monitored, it can compromise patient trust and confidentiality, which can result in negative health outcomes. Metadata protection can help prevent this by ensuring that patient information is kept private and that communication between doctors and patients remains confidential.

In addition to journalists and doctors, metadata protection is also important for protecting the privacy and security of other individuals, such as political dissidents, human rights activists, and individuals who may be at risk of surveillance or persecution by their government or other groups.

- [ ] **What are the limitations of existing e2e encrypted messengers.**

While end-to-end encrypted messengers offer a high level of security and privacy, there are some limitations to this technology that can impact the user experience.

One limitation is that new members joining a conversation will not have access to previous messages. This is because end-to-end encryption typically encrypts messages with a unique key that is only shared between the sender and recipient. When a new member joins a conversation, they do not have access to this key, so they cannot decrypt previous messages. This can make it difficult for new members to get up to speed on a conversation.

Furthermore, end-to-end encrypted messaging can be data-heavy on the client-side. Since all messages are stored on the client device and deleted from the server, this can require a lot of storage space on the user's device. This can be problematic for users with older devices with limited storage space. When the device is lost or destroyed the messages are lost as well. While this can be a security feature by itself, it can also be a painful experience for the user.

Another limitation is synchronisation problems, which can occur when a user is logged in to multiple devices. Since end-to-end encryption is device-specific, it can be difficult to synchronise messages across multiple devices while maintaining end-to-end encryption. This can result in messages being delivered to some devices but not others, or messages being marked as read on one device but not on others.

- [ ] **What is Discord, comparison to simple messaging apps.**

Discord is a communication platform designed for communities to connect and interact online. It allows users to join or create servers, which are essentially chat rooms or virtual spaces where users can chat via text, voice, or video, share files, and collaborate on projects.

Compared to simple messaging apps like SMS or Facebook Messenger, Discord offers a much richer and more interactive communication experience. Some key features of Discord include:

1. Servers: Discord allows users to join or create servers, which are essentially chat rooms or virtual spaces where users can chat via text, voice, or video.
2. Channels: Within servers, users can create different channels for different topics or purposes, which helps to keep conversations organized.
3. Voice and video chat: Discord allows users to make voice and video calls within servers, which can be useful for gaming, remote work, or socializing.
4. File sharing: Users can share files within servers, which can be useful for collaborative projects or sharing media.
5. Bots: Discord allows users to add bots to servers, which are essentially automated programs that can perform a variety of tasks, such as moderating chats, playing games, or streaming music.

Overall, Discord offers a much more feature-rich and interactive communication experience compared to simple messaging apps. It's designed to be a platform for communities and groups, rather than just one-to-one communication, and offers a range of tools and features to support that.

Big communities need some kind of administration to function properly and to ensure the a good experience for the members of that community. As an administrator on Discord, the user experience is focused on managing the server and creating a positive experience for members. Some key features and functions of the Discord administrator user experience include:

1. Server settings: Administrators can access and modify server settings, such as channel permissions, server name and icon, and server region.
2. User management: Administrators can manage users on the server, including adding and removing members, assigning roles and permissions, and moderating chats.
3. Channel management: Administrators can create and manage channels on the server, including text, voice, and video channels. They can also set permissions for each channel, allowing certain roles or users to access specific channels.
4. Bot integration: Administrators can add bots to the server, which can perform a variety of tasks, such as moderation, music streaming, or game integration.
5. Moderation: Administrators are responsible for moderating the server and ensuring that members adhere to community guidelines and rules. This includes managing chat, removing inappropriate content, and issuing warnings or bans as needed.

- [ ] **Characteristics of Discord.**

Discord like many other non e2e encrypted messaging platform allows new users of a group to have access to the historical conversation. There are various reasons why people would need a group chat application where a new member would have access to the previous conversation and pinned items. Some possible use cases could include:

1. Team collaboration: In a team collaboration setting, it may be important for new members to be able to access previous conversations to get up to speed on project updates, decisions made, and actions taken.

2. Educational settings: In an educational setting, such as an online course or study group, new students may need access to previous conversations to catch up on discussions, ask questions, and contribute to ongoing conversations.
3. Community groups: In a community group, such as a neighborhood watch or social club, new members may need access to previous conversations to stay informed about group activities, discussions, and decisions.
4. Support groups: In a support group, such as a mental health or addiction recovery group, new members may need access to previous conversations to read about the experiences and advice of others in the group.
5. Business meetings: In a business meeting setting, new members may need access to previous conversations to review meeting agendas, organisation details, and action items.
6. File Sharing: In many cases users want to exchange files like images and videos. New members would be able to see these files even if they arrive after the files have been sent to the room.
7. Asynchronous addition of members: A moderator creates a room and wants to make an announcement to all present and future members of that room. Whenever a new member joins the room, they would be able to see that announcement making the moderator’s work a lot more effortless.

In each of these use cases, having access to previous conversations can help new members get up to speed quickly and stay informed about ongoing discussions and decisions.

- [ ] **What is an encrypted Discord, why is it useful.**

In end-to-end encrypted messaging applications, it can be advantageous for new members to not have access to prior conversations, as this may raise privacy concerns among existing group members. In such scenarios, members may only trust the individuals present in the group at that moment in time, rather than any new members who may join in the future. While this approach can enhance security, it may also create a hurdle to the user experience.

Below are some use cases for an end-to-end encrypted group chat application where a new member would have access to the previous conversation:

1. Legal teams: In a legal setting, it may be necessary to share confidential information and documents with a new member who has recently joined the team. End-to-end encryption ensures that the conversations and files are secure, while access to previous conversations allows the new member to get up to speed quickly.
2. Research groups: In a research group, members may need to discuss sensitive information related to ongoing projects. End-to-end encryption ensures that the conversations are secure, while access to previous conversations allows new members to understand the progress and findings of previous discussions.
3. Political campaigns: In a political campaign, team members may need to discuss sensitive information related to campaign strategy, fundraising, and voter outreach. End-to-end encryption ensures that the conversations are secure, while access to previous conversations allows new team members to understand the progress and decisions made by the campaign.
4. Journalistic sources: In a journalistic setting, sources may need to share confidential information with a journalist and team. End-to-end encryption ensures that the conversations are secure, while access to previous conversations allows new team members to understand the background and context of previous discussions.
5. Non-profit organizations: In a non-profit organization, team members may need to discuss sensitive information related to donors, fundraising, and outreach efforts. End-to-end encryption ensures that the conversations are secure, while access to previous conversations allows new team members to understand the progress and decisions made by the organization.

In contrast to local storage-based messaging applications like Signal, the e2e encrypted Discord client application is relatively lightweight due to the fact that conversations are stored on the server, with the client only downloading what is required. This is useful especially in large communities with more than 300 members, media-rich content such as images and videos can make conversations unwieldy. Additionally, a user may not be interested in a particular channel from the community room, but still desires the ability to access it if needed. E2E applications on the market have a cap on the amount of users allowed in a group chat. This is partially due to the high demand of local-storage these apps need to function. The proposed architecture of the e2e encrypted Discord application does not need to have a limit on the amount of members of a room. However it is important to note that the more users there are in the room, the more risk there is for the shared keys to be leaked. We are going to delve into that in the following sections.

☐ How does Signal work and what are its security features

Signal provides end-to-end encrypted messaging for its users, but how did Open Whisper Systems design their protocol to accomplish this feat? In order to answer this question, this section will serve as a basis for learning the cryptographic primitives used in the Signal Protocol. A cryptographic primitive is a low-level algorithm, commonly used as a building block to construct more complex cryptographic protocols. The Signal Protocol even uses custom-made cryptographic algorithms to accomplish the security goals of the application, which will be covered in detail later in this section.

Before defining the cryptographic primitives used within the Signal Protocol, it is vital that an understanding is built of what these building blocks are used to accomplish. Three well known security goals of any secure messaging application include confidentiality, message authenticity and origin integrity.

• **Confidentiality** is the guarantee that messages sent between the two or more parties can only be viewed by authorized individuals, and that non-intended recipients cannot view the message — this accomplished by the use of encryption and decryption schemes. 1 https://signal.org/blog/private-groups/ 2

• **Origin Integrity** is a security goal which guarantees when a user Alice receives a message from another user Bob, Alice knows this message did in fact comes from Bob – this can be accomplished by the use of cryptographic signatures sent between two parties.

• **Message Authenticity** is the guarantee that a message between two parties has not been altered in any way by an unauthorized party – this can be accomplished by the use of message authentication codes (MACs).

These security properties are vital to any encrypted messaging application, without these adversaries may intercept and read messages intended for other users, impersonate another user, and/or edit the contents of messages intended for other users.

Based on previous work [6], which examines the security guarantees provided by various other encrypted messaging applications, we find that the Signal Protocol additionally provides forward secrecy, future secrecy, participant consistency, destination validation, causality preservation, message unlinkability, message repudiation and participant repudiation. We will base our definitions of each property by the descriptions provided by Unger et al. [6], as well as Schliep and Hopper [10] as follows:

• **Forward secrecy** is a feature of key agreement protocols which prevents an adversary who compromises the message keys of a target user from decrypting any past messages. • Future secrecy is another feature of key agreement protocols which prevents an adversary who compromises the message keys of a target user from decrypting any future messages in the conversation to some extent.

• **Participant consistency** is a security trait derived from the encryption and authentication of the message transcripts from a given conversation. This ensures that each participant in a communication channel has the same list of members in the channel. In the context of a conversation, this ensures that the set of members in this channel is identical for each user.

• **Participant repudiation (or Deniability)** relies on the encryption and authentication of message transcripts, as well as the fact that Signal servers do not store conversation metadata. As a result, any participant is able to deny their participation in a conversation as long as private keys are not compromised.

• **Message unlinkability** is similar to the previously mentioned Participant repudiation / Deniability trait. To define this trait, let's assume that a judge has been convinced that a participant authored a message. Even with this assumption, having message unlinkability ensures that this fact does not provide evidence that they authored other messages within the conversation.

• **Message repudiation** is similar to Message unlinkability. For a given conversation and all its encryption/decryption keys, a user can deny that they authored a particular message. • Destination validation is the assurance that a user can verify that they were in fact the intended recipient of a given message from another user.

• **Causality preservation** is a side effect to the implementation of the Double Ratchet algorithm within the Signal Protocol – if a message is delayed and another more-recent message is received before the original arrives to its destination, implementations can opt to wait for the first message to be received before decrypting the other.

The properties mentioned above describe the protections that are in place which safeguard Signal’s users. To understand how these safeguards are implemented, we will take an in-depth look at the algorithms which support Signal’s key distribution and message encryption functions.

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/e5973c79-39fa-4693-b54d-7b666128879d/JansenMatthewL2020.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/32550c64-e642-42e6-b07e-d4237962ed6e/z00b_2019_thesis_dion_van_dam_2019_eerder.pdf

Specifications >> The X3DH Key Agreement Protocol

This document describes the "X3DH" (or "Extended Triple Diffie-Hellman") key agreement protocol.

https://signal.org/docs/specifications/x3dh/#deniability

Private Group Messaging

One of the major features we introduced in the TextSecure v2 release was private group chat. We believe that group chat is an important feature for encrypted communications projects, so

https://signal.org/blog/private-groups/

☐ **Encryption Schemes and Tools**

- [ ] **Symmetric Key Cryptography**
- [ ] **Asymmetric Key Cryptography**
- [ ] **PGP - X3DH**

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/19b2cc9a-dabf-4a2a-8c38-e6b7dd3d5f82/x3dh.pdf

- [ ] **Forward Secrecy, Double Ratchet**

Specifications >> The Double Ratchet Algorithm

The Double Ratchet algorithm is used by two parties to exchange encrypted messages based on a shared secret key.

https://signal.org/docs/specifications/doubleratchet/

Signal

- [ ] **What are the security features of e2e discord, comparison to signal, security balance with usability.**

The aim of this study is to develop an end-to-end (e2e) encrypted platform that incorporates features similar to those found in Discord, most notable is the ability for new members to see older messages. In comparison, the Signal Messenger because it incorporates the security feature of Forward Secrecy, does not provide that ability to the users. Like in many cases in computer science, to provide more complex features some things must be sacrificed. In the context of encrypted applications we can find a balance between security and usability.

The proposed platform utilizes shared symmetric key encryption to encrypt messages sent to a group, which are then stored on a server. Members of a group use one or many shared symmetric keys to encrypt all the messages in the room. This design enables newly invited members to access previously uploaded posts, providing a user experience akin to social media sites. The platform's architecture also ensures negligible synchronization issues as each message is uniquely stored within the room. However, this architecture lacks certain cryptographic features, such as **forward secrecy and non-repudiation**, which are available in the Signal Protocol. Furthermore, while persistent storage offers some benefits, it also has some security vulnerabilities.

Developing more complex e2e encrypted applications like this while maintaining the high level of security and privacy provided by the Signal Protocol can be challenging. As a result, a balance between usability and security may be an acceptable solution for some users.

To enable secure communication, an invitation mechanism is in place. The service provider must never have access to the shared keys. To accomplish that, users generate a personal public-private key pair, similar to the UID in the Signal Protocol. With this asymmetric cryptographic tool users only need to exchange identities (public keys) in another channel such as Signal or real life. The inviter takes their friend's public key and encrypts the shared symmetric key of the room, sending it to the server. This process ensures that only the person with the corresponding private key can decrypt the message, which is similar to PGP. A suitable analogy to this process is the way users invite each other on Discord, where a friend sends their account name, and the recipient simply copies and pastes it into the Discord app. This mechanism ensures that the shared symmetric key/keys of the room exist decrypted only on end-user devices and not in potentially unsecured channels.

The Signal app avoids the above process of public key exchange by storing a mapping between phone numbers and UIDs on the server. As a result, Signal users must initially trust the Signal server when sending the first message. A potential malicious server could change the mapping to a UID they own and decrypt messages, impersonating the receiver in a Man in the Middle attack. This flaw arises from the members of the conversation not verifying each other's identities beforehand. Users of the Signal app can verify the security of the channel afterwards if desired.

- ☐ **Trustless System, cryptographic verification.**
  - ☐ **Metadata Protection, Limitations of centralized systems**
  - ☐ **ZK-proofs, Anonymous Credentials**

Technology preview: Sealed sender for Signal

In addition to the end-to-end encryption that protects every Signal message, the Signal service is designed to minimize the data that is retained about Signal users. By design, it

https://signal.org/blog/sealed-sender/

Technology Preview: Signal Private Group System

Groups are inherently social, and Signal is a social app. Whether you're planning a surprise party, discussing last night's book club meeting, exchanging photos with your

https://signal.org/blog/signal-private-group-system/

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/6f690be5-0647-4b53-bf4e-dfecb8456b42/signal_private_group_system.pdf

- [ ] **Blockchain Data Structure comparison to User Lists in Signal**