1 Analysis

This chapter consists of two parts. The first part will provide an evaluation of the Matrix security model and relies on the paper *SoK: Secure Messaging* [**?**] and *The Olm Cryptographic Review* by NCC Group [**?**].

The second part provides a preliminary analysis of the IFC tools, the selection of Paragon and the rationale behind it, and a further analysis of the selected tool Paragon.

1.1 Evaluation of Matrix security model

The security of matrix will be evaluated in the context of secure messaging. An evaluation framework has been proposed in the paper *SoK: Secure messaging* which the evaluation will be loosely based on.

The evaluation framework covers several areas with *conversation security* being the most relevant for this evaluation. The area *conversation security* describes three categories; *Security and Privacy*, *Adoption*, and *Group Chat*. Obviously the most relevant category for the evaluation is *Security and Privacy*

Matrix provides end-to-end encryption by using the Olm and Megolm library with the former being an implementation of the Double Ratchet algorithm also known as the Signal Protocol, and the latter being the algorithm used for group chat.

Olm is used for securely exchanging message keys/session keys during group chat and is vital part of the end-to-end encryption in Matrix.

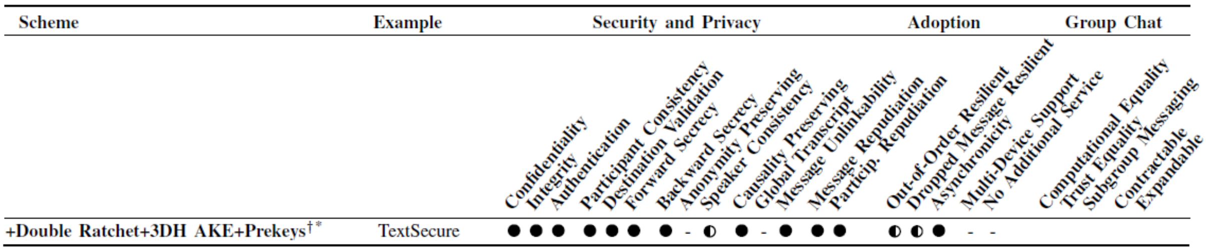
Before the Matrix protocol is evaluated the Signal Protocol will be considered.

1.1.1 Signal Protocol

Olm is an implementation of the Signal Protocol. The Signal Protocol is a messaging protocol with end-to-end encryption between two devices.

Section xx provides a list of security properties relevant for *conversation security.* These security properties is used for evaluating a secure messaging protocol such as the Signal Protocol.

The table below shows an evaluation of the Signal Protocol (previously known as TextSecure) [**?**].



Scheme	Example	Security and Privacy														Adoption					Group Chat				
		Confidentiality	Integrity	Authentication	Participant Consistency	Destination Validation	Forward Secrecy	Backward Secrecy	Anonymity Preserving	Speaker Consistency	Causality Preserving	Global Transcript	Message Unlinkability	Message Repudiation	Particip. Repudiation	Out-of-Order Resilient	Dropped Message Resilient	Asynchronicity	Multi-Device Support	No Additional Service	Computational Equality	Trust Equality	Subgroup Messaging	Contractable	Expandable
+Double Ratchet+3DH AKE+Prekeys$^{\dagger *}$	TextSecure	●	●	●	●	●	●	●	-	◐	●	-	●	●	●	◐	◐	●	-	-					

Figure 1.1: Evaluation of Signal (TextSecure) [?].

Confidentiality When a message is sent using the Signal Protocol then only the intended recipient can read the message. The senders sending ratchet and receivers receiving ratchet will derive the same message key hence only the two parties will be able to encrypt the messages.

Integrity The receiver will only accept a message if it is successfully decrypted hence if in transit a message was modified then the message would be rejected.

Authentication The decryption of a message also gives authentication guarantees since only the intended recipient could compute the message key.

Forward secrecy The symmetric ratchet ensures forward secrecy. If a chain session key is compromised then the previous keys can not be generated since the ratchet is one way cryptographic hash function hence secrecy is provided for all previous send messages.

Backward secrecy Diffie-Hellman ratchet have the self-healing property and will generate a new chain session key for the symmetric ratchet hence if a chain key is compromised then secrecy for future messages is still provided because a new chain ratchet key will be generated.

Anonymity preserving Anonymity preservation is lost in the Signal Protocol since the initial key agreement requires long-term public keys hence making them observable during Triple-DH. However ***participant consistency*** is provided by Triple-DH [?].

Speaker consistency This property is partially provided through the key evolution of the ratchets. If a message is dropped then it is not possible to generate message keys for future messages. This also makes the protocol partially have the property ***Dropped message resilence***. It will also not go unnoticed if a message is received out of order since this will result in the message's key being an unexpected key. Hence the recipient have to store expired keys to decrypt delayed messages. This makes the property ***Out-of-order resilient*** only partially provided [?].

Global transcript In an asynchrounous messaging protocol there is no global transcript. Both participants have to be online to receive messages hence the participants will not have all the messages if one of them is offline. This is a result of having the ***Asynchronicity*** property.

Deniability properties Since the ratchet session keys are used for encrypting messages and not the long-term public keys the properties ***Message unlinkability*** and ***Message repudiation*** are provided.

Other properties

- ***Participant repudiation***. Triple-DH achieves full participant repudiation since anyone can forge a transcript between any two participants [**?**].
- ***Destination validation***. The Deffie-Hellman ratchet provides this property since the recipients public key is used to generate the chain key [**?**].

The evaluation shows that several security properties are provided with the important ones being confidentiality, integrity, authentication, forward secrecy, backward secrecy.

Furthermore a formal analysis have been made on the Signal Protocol that proves the protocol is free from any major flaws and it satisfy the following security properties; confidentiality, authentication and secrecy [**?**].

1.1.1.1 Application variants

The Olm library used by Matrix is a variant of the Double Ratchet algorithm. The custom variants invites important changes in need to be analyzed independently. WhatsApp variant of the protocol has a retransmission mechanism which is vulnerable. The further evaluation relies upon the the security assessment on Matrix.

1.1.2 Matrix protocol

1.1.2.1 Group chat

The Double Ratchet algorithm is meant for one-to-one chatting and is not practical for group chatting.

For direct conversation Matrix uses Olm library which is an implementation of the Double Ratchet algorithm. For group chats the Megolm library is used.

It is more difficult to evaluate the Matrix protocol because they as well do not specify any detailed security goal and there is at this point of writing no formaæ analysis of their implementation of the Double Ratchet algorithm.

List problems with group chat

- Lack of post compromise security

There is an tradeoff on security and usability and the security is decreased for group chats.

As of this moment of writing there exist no implemented protocol that solves these issues in group chat. However recent research has proposed solutions with early implementations for these problems with IETF leading the research on the standard on *Messaging Layer Security*. Matrix has expressed awareness of the protocol and a possibility of adaption in the future.

1.1.2.2 Olm

Olm is an implementation of the Double Ratchet algorithm and plays a major role in the Megolm library which is used by Matrix group messaging.

The security assessment provides a review of the Olm and Megolm libraries. However there exist no work that verifies if Olm correctly implements the Double Ratchet described earlier.

Vulnerabilities found in the security assessment.

Unknown Key Share attack

In this variant of the unknown key-share (OKs) attack, an attacker will allow highly targeted, known messages to be sent to Bob. In this scenario, Bob will still believe he is talking with Alice. Here, two parties (Donald and Mallory), who may be the same person, will collude against Alice in a group chat situation (Megolm). Donald will be performing the unknown key-share attack. Mallory will be an instigator (attempting to elicit messages from Alice that will later be sent to Bob) who will be able to read the contents of the group chat.

1.1.2.3 Megolm

Conceptually, when you first send a message in an encrypted room, your Riot client generates a random key to encrypt your message, sends the encrypted message to the server, and then sends the decryption key to all the devices in the room that should be allowed to decrypt the message. Of course, the decryption key is sent encrypted (based on the device's unique key, which you verified above) so that it cannot be intercepted. The recipient then fetches the message decryption key and the encrypted message and decrypts the message.

In order to avoid having to re-send decryption keys to every device for every message you send, Matrix's encryption system includes a method for generating a new key based on an old key. So for the next message you send, your Riot client will use that method on your previous encryption key to generate a new key, and the recipients will use the same method and generate the same key, so that when you send a message encrypted using the new key, the recipients can decrypt the message without any extra key exchange. The new key will only need to be sent to any new devices that showed up in between when the first message was sent and when the second message was sent.

Riot will occasionally start from scratch, generating a new random key and sending it to all the devices in the room. This happens, for example, whenever someone leaves a room, after you have sent a certain number of messages, or after a certain amount of time.

Message Replays

A message can be decrypted successfully multiple times. This means that an attacker can re-send a copy of an old message, and the recipient will treat it as a new message.

To mitigate this it is recommended that applications track the ratchet indices they have received and that they reject messages with a ratchet index that they have already decrypted.

Lack of Transcript Consistency

In a group conversation, there is no guarantee that all recipients have received the same messages. For example, if Alice is in a conversation with Bob and Charlie, she could send different messages to Bob and Charlie, or could send some messages to Bob but not Charlie, or vice versa.

Solving this is, in general, a hard problem, particularly in a protocol which does not guarantee in-order message delivery. For now it remains the subject of future research.

Lack of Backward Secrecy

Once the key to a Megolm session is compromised, the attacker can decrypt any future messages sent via that session.

In order to mitigate this, the application should ensure that Megolm sessions are not used indefinitely. Instead it should periodically start a new session, with new keys shared over a secure channel.

Partial Forward Secrecy

Each recipient maintains a record of the ratchet value which allows them to decrypt any messages sent in the session after the corresponding point in the conversation. If this value is compromised, an attacker can similarly decrypt those past messages.

To mitigate this issue, the application should offer the user the option to discard historical conversations, by winding forward any stored ratchet values, or discarding sessions altogether.

Dependency on secure channel for key exchange

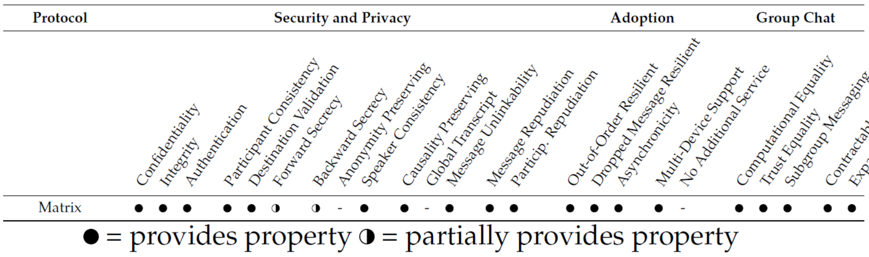
The design of the Megolm ratchet relies on the availability of a secure peer-to-peer channel for the exchange of session keys. Any vulnerabilities in the underlying channel are likely to be amplified when applied to Megolm session setup.

For example, if the peer-to-peer channel is vulnerable to an unknown key-share attack, the entire Megolm session become similarly vulnerable. For example: Alice starts a group chat with Eve, and shares the session keys with Eve. Eve uses the unknown key-share attack to forward the session keys to Bob, who believes Alice is starting the session with him. Eve then forwards messages from the Megolm session to Bob, who again believes they are coming from Alice. Provided the peer-to-peer channel is not vulnerable to this attack, Bob will realise that the key-sharing message was forwarded by Eve, and can treat the Megolm session as a forgery.

A second example: if the peer-to-peer channel is vulnerable to a replay attack, this can be extended to entire Megolm sessions.

1.1.2.4 Evaluation

The evaluation framework in the paper *SoK: Secure Messaging* defines *Conversation Security* as a problem area which contains the following main groups; *Security and Privacy Featues*, *Adoption*, and *Group Chat*.



Protocol	Security and Privacy														Adoption					Group Chat				
	Confidentiality	Integrity	Authentication	Participant Consistency	Destination Validation	Forward Secrecy	Backward Secrecy	Anonymity Preserving	Speaker Consistency	Causality Preserving	Global Transcript	Message Unlinkability	Message Repudiation	Particip. Repudiation	Out-of-Order Resilient	Dropped Message Resilient	Asynchronicity	Multi-Device Support	No Additional Service	Computational Equality	Trust Equality	Subgroup Messaging	Contractable	Expandable
Matrix	●	●	●	●	●	◑	◑	-	●	●	-	●	●	●	●	●	●	●	-	●	●	●	●	●

● = provides property ◑ = partially provides property

Figure 1.2: Evaluation of Matrix Security.

Confidentiality When a message is sent using the Signal Protocol then only the intended recipient can read the message. The senders sending ratchet and receivers receiving ratchet will derive the same message key hence only the two parties will be able to encrypt the messages.

Integrity The receiver will only accept a message if it is successfully decrypted hence if in transit a message was modified then the message would be rejected.

Authentication The decryption of a message also gives authentication guarantees since only the intended recipient could compute the message key.

Forward secrecy The symmetric ratchet ensures forward secrecy. If a chain session key is compromised then the previous keys can not be generated since the ratchet is one way cryptographic hash function hence secrecy is provided for all previous send messages.

Backward secrecy Diffie-Hellman ratchet have the self-healing property and will generate a new chain session key for the symmetric ratchet hence if a chain key is compromised then secrecy for future messages is still provided because a new chain ratchet key will be generated.

Anonymity preserving Anonymity preservation is lost in the Signal Protocol since the initial key agreement requires long-term public keys hence making them observable during Triple-DH. However ***participant consistency*** is provided by Triple-DH [**?**].

Speaker consistency This property is partially provided through the key evolution of the ratchets. If a message is dropped then it is not possible to generate message keys for future messages. This also makes the protocol partially have the property ***Dropped message resilence***. It will also not go unnoticed if a message is received out of order since this will result in the message's key being an unexpected key. Hence the recipient have to store expired keys to decrypt delayed messages. This makes the property ***Out-of-order resilient*** only partially provided [**?**].

Global transcript In an asynchrounous messaging protocol there is no global transcript. Both participants have to be online to receive messages hence the participants will not have all the messages if one of them is offline. This is a result of having the ***Asynchronicity*** property.

Deniability properties Since the ratchet session keys are used for encrypting messages and not the long-term public keys the properties ***Message unlinkability*** and ***Message repudiation*** are provided.

Other properties

- ***Participant repudiation***. Triple-DH achieves full participant repudiation since anyone can forge a transcript between any two participants [**?**].
- ***Destination validation***. The Deffie-Hellman ratchet provides this property since the recipients public key is used to generate the chain key [**?**].

1.1.3 End-to-end security

It can be argued that some of the Matrix shortcomings can be configured on the application layer. This puts a lot of responsibility on the application and that it is configured correctly.

Even if the application using Matrix is configured correctly and give the best guarantee of forward and backward secrecy this not the end of security.

Dynamic policies?

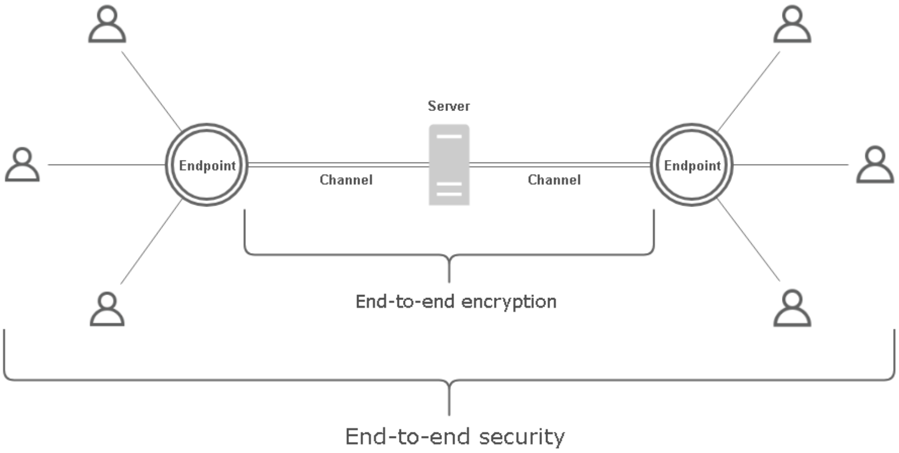


Figure 1.3: End-to-end security.

Recently two technical directors from the Government Communication Headquarters of United Kingdom released an essay arguing that the software vendors could grant access to group chat by inserting the government as a hidden silent participant in a chat hence not weakening the encryption [**?**].

Explain how some of the shortcomings in Matrix can be solved with IFC.The initial state ratchet is never ratcheted?

1.1.4 Summary