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*

1.1.1 Threat model

For secure messaging the evaluation framework defines a threat model with three types of adversaries. Note that an adversary can be of several types:

- *Local adversary:* The adversary is in control of the local network.
- *Global adversary:* The adversary is in control of great portions of the Internet
- *Service providers:* A potential adversary for messaging systems with centralized infrastructure.

In the messaging system the adversary may be a participant with the following properties:

- An adversary can start a conversation.
- An adversary can send messages.
- An adversary can perform any other action that a participant is capable of.

Furthermore it is assumed that the system's endpoints are secure [**?**].
This evaluation will inherit the described threat model.

1.1.2 The Signal Protocol

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. The Signal Protocol is described in section xx.

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[†*]	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 have the property ***Causality Preserving*** and 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 [**?**].

Application variants

The Signal Protocol is a secure messaging protocol and have been extensively studied including proof that the standard security properties are assured.

The Olm library used by Matrix is a variant of the Signal Protocol. There is no implementation analysis of the Olm library hence there is no guarantee that all the security properties defined in xx is inherited by Olm. Nevertheless it is assumed that Olm inherits the above properties.

The further evaluation relies upon the the security assessment of Matrix.

1.1.3 Matrix protocol

As described in section xx *rooms* are a fundamental part of Matrix' architecture. There can be multiple participants in a room hence the support for secure group conversation is required.

Olm (and the Signal Protocol it is based on) is ideally meant for two party communication. Group conversation could be supported with a näive variant of Olm. In a group with N participants each participant would establish a secure Olm session with every other participant. When a message is send each message would then have to be encrypted N times. This solution would scale poorly if N was a large number. This was the motivation for introducing Megolm.

Megolm

Megolm is a multicast encryption solution [**?**].Each sender has a sender ratchet (Megolm Ratchet). Each recipient has a corresponding receiving ratchet for each sender. So if there are N participants in a group then each participant will have N-1 receiving ratchets. Figure 1.2 illustrates the setup with three participants.

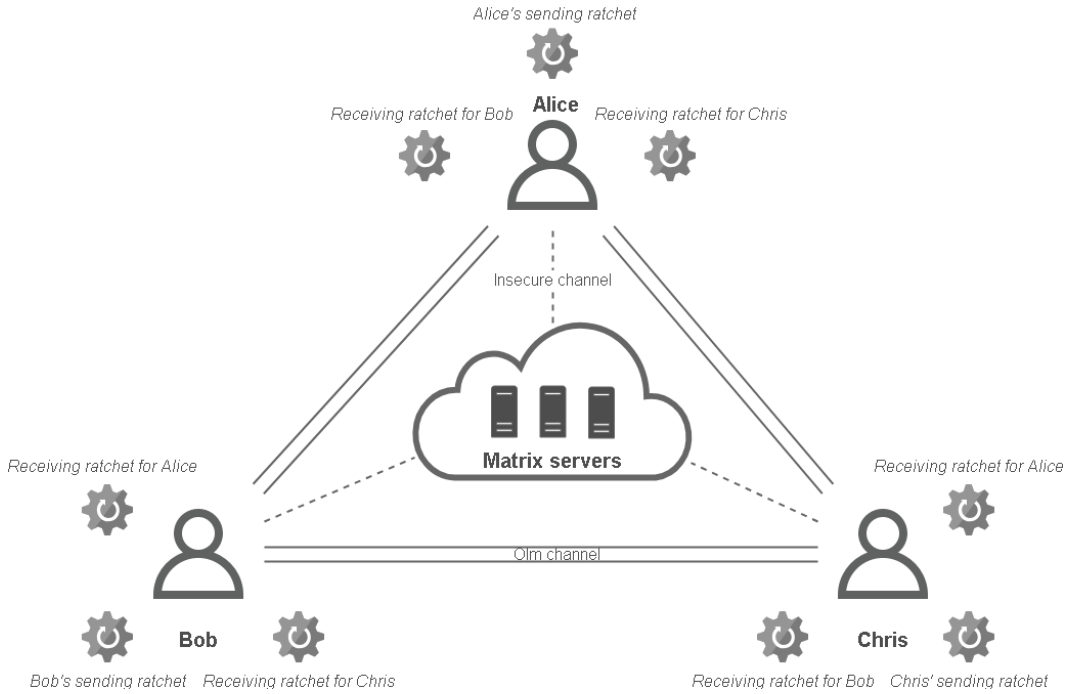

Figure 1.2: Conceptual model of Megolm with three participants.

When a session is started a sender will send his initial ratchet key to each recipient, so that the sender ratchet and each recipients ratchet are in sync. This key exchange happens over a secure communication channel (Olm). Furthermore there is send N-1 initial messages when a session is initiated. Until a new session is started no further session keys are exchanged and the corresponding message keys are generated by incrementing the ratchet.

When a sender sends a message a message key is generated from the ratchet key and the message is encrypted using that message key. The message will be signed so the recipient will know which sender the message is from and which ratchet to utilize. The message is then send to the server which relays the message to all recipients over an insecure channel. When they receive the message the same message key is generated using the corresponding receiver ratchet and the message is decrypted.

When a new participant joins the latest ratchet key would then be shared by each participant over Olm (or an earlier one if he should have access to historical conversation).

When a participant leaves a new session would be initiated yielding in refreshing the ratchet keys hence not making it possible for that ex-participant to decrypt any further messages.

The Matrix Protocol will be evaluated in the context of Megolm. The evaluation of the Matrix protocol heavily relies on the security assessment by NCC.

1.1.3.1 Evaluation

The Matrix Protocol provides several security properties shown in the table xx.

It is worth mentioning that there is a trade-off between security and usability which must be decided at application layer. The most secure configuration would come at the cost of usability and performance.

- *Usability.* From a users point of view it would be nice to have the possibility to load historical conversation instead of having to keep full history locally. Matrix supports multiple devices and if a participant adds another device at some later point it makes sense to load the participants historical conversation into the device. From a security perspective this would mean that the *initial ratchet state* is stored and is send to the new device so every message key can be generated. This certainly goes against the principle of forward secrecy. The most secure configuration would not store the *initial ratchet state* hence satisfy forward secrecy thus disable the described usability feature [**?**] [**?**].

- *Performance.* When a megolm session is initialized there is an initial burst of messages to exchange the initial ratchet key which is then stored in a *initial ratchet state* value at each recipient. If this key is compromised then any future key can be generated for that session. To satisfy backward secrecy this would mean initiating a new session for each message which would trigger a burst of messages to exchange the ratchet key [**?**] [**?**]. This would scale poorly for a large group or when sending large-sized messages.

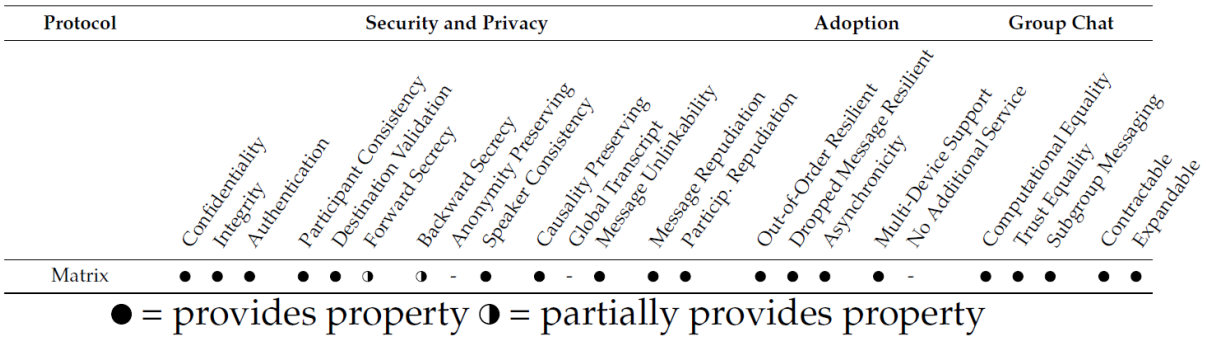

Figure 1.3: Evaluation of Matrix Security.

Some of the security properties in the table are briefly examined.

Confidentiality When a message is send it is encrypted and can only be decrypted by the intended recipients who has the corresponding ratchet session key received over an Olm channel.

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.

Forward secrecy Each participant keeps a *initial ratchet state* which holds the earliest ratchet session key for a session. This clearly violates forward secrecy since every message can be decrypted if the *initial ratchet state* value is compromised. However it is a deliberate trade-off for usability to enable historical conversation and storing the value is optional. Since this is an optional feature the forward secrecy is partially provided [**?**].

Backward secrecy If a ratchet key is compromised then an adversary can generate every message key from that point on hence intercept any message that sender sends to the group. This can be prevented strictly by starting a new session with every send message however it would not be possible to keep conversation history (only locally when data is encrypted). Hence the property is only partially provided [**?**].

Speaker consistency There is no guarantee for speaker consistency. A well known problem of multi-cast encryption group chat is transcript inconsistency. A sender may send different messages to different recipients. However it requires that the server is in collusion with the sender. This also applies to ***Causality preserving*** [**?**].

Other properties

The multi-cast encryption design does not provide *participant consistency* [**?**].

The properties *Dropped message resilence* and *Out-of-order resilient* are provided by keeping track of ratchet indices.

Several properties are inherited from the secure key exchanging channel provided by Olm while other properties are inherited because of asynchronicity of the Megolm protocol.

- *Authentication* is provided by Olm since the ratchet session key is send to the recipient through an Olm channel or else the message key could not be derived.
- *Destination validation.* The ratchet session key is exchanged over a secure Olm channel hence only the intended recipient could decrypt it.
- *Anonymity preserving* is not provided since Olm requires the long-term public key in the initial key exchange.
- *Global transcript* is also not provided because of the asynchronous nature of the Megolm protocol.
- *Asynchronicity* is obviously provided.
- *Deniability* properties are inherited from Olm as well.

All properties related to group chat are also provided. Although they are additional features and not related to security.

Other findings

Message Replays Matrix allows decryption of a message multiple times hence it is vulnerable to replay attacks. Replay attacks are handled at the application layer. Whenever a message is decrypted a message index is generated and stored. If the exact message is decrypted again the same message index will be generated and can be compared to the stored message index making the replayed message invalid.

Unknown Key Share attack A vulnerability found with a high risk in Megolm. The vulnerability is inherited from Olm and occurs after the initial message in Triple-DH. The attack is described in section xx.

The vulnerability has been mitigated at the application layer by providing a unique identifier for the sender and receiver into each message and then checking the values when decrypted [**?**].

Recent research

The way backward secrecy would be provided in Matrix is computationally expensive. 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.

1.1.4 End-to-end security

Section xx describes Matrix long-term goal as being a generic HTTP messaging API. It could be utilized for any kind of data exchange in a system or between multiple systems.

A system using the Matrix Protocol for exchanging data would benefit from the security properties found in the evaluation yet the system-wide security or end-to-end security would be incomplete an further measures must be taken. Such system might demand confidentiality and integrity throughout the system yet the system as a whole would have a different threat model than the one described for a secure messaging system hence no guarantee of confidentiality or integrity beyond the endpoints in end-to-end encryption.

The following figure depicts how end-to-end encryption might be inadequate in such system.

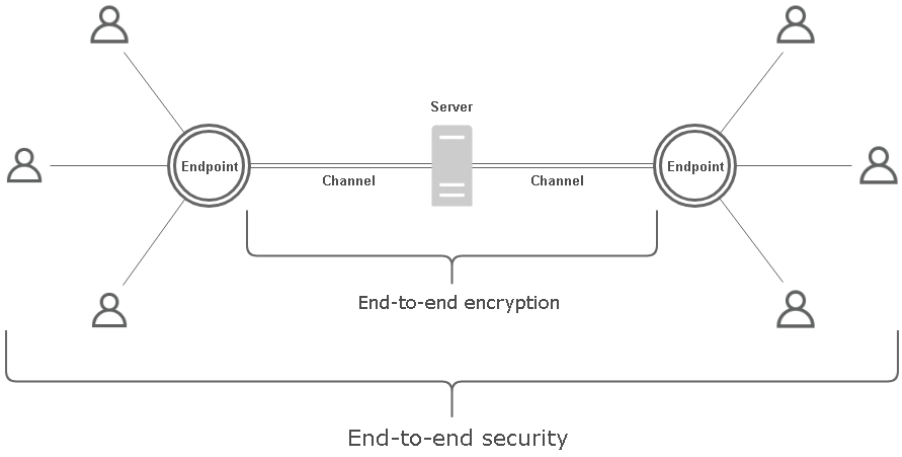

Figure 1.4: End-to-end security.

As the system depicts there might be several principals accessing the endpoint. Each principal could retrieve some information possibly protected with access control. Assume that the information resting at the endpoint is of confidential nature access could still be granted with no respect of the confidentially of that information. There clearly lack a mechanism of specifying what information is confidential or public and where it may flow under what conditions.

Matrix identifies IoT as another use case. A person can have several devices for health tracking, entertainment and so on. The data from the devices are send to vendors - a device might send data to several vendors. Ultimately this give a fragmentation of the person's own data with it being placed at several vendors data back-end. Matrix proposes a solution where all the device data for a person is synchronized and persisted on Matrix. Vendors would be connected to Matrix. This is depicted in figure 1.5 below.

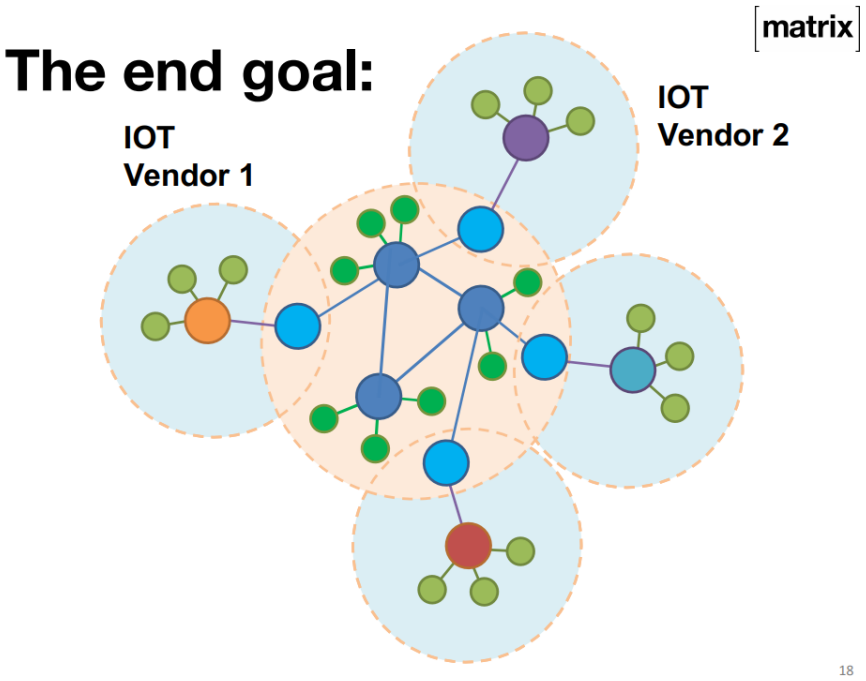

Figure 1.5: End goal for Matrix IoT

The data flowing from the sensors to Matrix might be of sensitive nature or the owner might only allow some data to flow to some vendors under specific conditions. This issue is not addressed by Matrix.

1.1.5 Summary

In this section an evaluation of Matrix security was presented. Several security properties are a part of Matrix security model.

End-to-end encryption is not the end of security. Vulnerable endpoints can still leak and can be secured with Information flow control.

The next session presents the analysis of Information-Flow Control tools.