IT University of Copenhagen

(master thesis)

Information-Flow Secure Programming on Matrix: A Case Study

Authors:
Ans Uddin anud@itu.dk

Supervisor:
Willard Rafnsson

Co-supervisor:
Carsten Schürmann

March, 2019

Abstract

Acknowledgements

Contents

List of Tables

List of Figures

1 Introduction

1.1 Data privacy and protection

With GDPR becoming effective in 2018 the focus on data privacy is at its peak. Privacy violation is when sensitive data is exposed to unauthorized actors[3]. OWASP top ten ranks *sensitive data exposure* as 3rd biggest security threat[16].

Recent cases of data leakage has put more attention on data privacy and protection. Some cases are due to poor security measures and could arguably have been prevented. Examples of cases are:

- The infamous Facebook - Cambridge Analytica scandal. Third parties were able to collect data through Facebook Login API.
- Google Plus leak. 500.000 users private data was exposed to third parties through APIs[6].
- Medicaid leak. A medical assistant had accessed patients' health records and exchanged mails with another employee containing the patients' private data[9].

The cases above failed to achieve end-to-end security and the improper handling of sensitive data could have been prevented with appropriate security policies and enforcement technique that enforces these policies.

There is more awareness on how applications deal with data. This add extra concern to the programmer and the application about how sensitive data is handled and protected.

The well-known security enforcement techniques like access controls, firewalls and encryption are inadequate alone and does not ensure end-to-end security[19].

1.2 Information Flow Control

There exist useful security enforcement mechanisms for protecting confidential information such as firewalls, encryption and access control. However, these mechanisms each have their drawbacks.

- *Access* control prevents unauthorized access to information but once access is granted there is no guarantee how that confidential information is handled.

- *Firewall* limits communication from the outside hence isolate and protect information. Yet the firewall have no way of telling if the communication going through violates confidentiality.
- *Encryption* secures information on a channel with only the endpoints being able to access that information. However there is no assurance that once the data is decrypted that the confidentiality of that information is ensured.

The mechanisms mentioned above all have in common that they lack control of how the information flows. Information-flow security aims at protecting confidentiality and integrity of information by enforcing security policies. Information-Flow Control allows the programmer to define and enforce policies in a language-based way[19].

1.3 Matrix

Matrix is an open standard protocol for messaging over HTTP and synchronizing data. Matrix provides secure real-time communication over a decentralized federated network. Matrix secures data by providing end-to-end encryption.

Matrix cover use cases such as instant messaging, VoIP, Internet of Things communication and is generally applicable anywhere for subscribing and publishing data over standard HTTP API.

The fragmentation of IP communication is the problem Matrix essentially wants to solve. Making calls and messages between users needless of which app they use. However they define their longer term goal as *"to act as a generic HTTP messaging and data synchronisation protocol for the whole web"*[8].

1.4 The case study

The goal of the case study is to make secure implementation of a prototype using Information-Flow Control. The case study will use Matrix as the communication channel and strengthen the security at the endpoints using IFC.

1.4.1 Journal system

The prototype implements a journal system and is loosely based on the Danish E-journal system.

Medical privacy is a well-known issue[18]. Sensitive data about patients needs to be handled carefully. In Denmark patients have access to their medical records through E-journal[4]. A patient's journal on E-journal is available for up to 90.000 different medical employees[1].

There are clear policies about who and under what conditions should access a journal. It is legally required that an employee accessing the journal must have the patient in care and that the lookup must be relevant for the employee. Safety measures have been applied through logging and audit trails with random sampling checks however they do not prevent access to journals. Any medical employee can access the patient journal and even if prevention mechanism were

established there would be no limitation to what a medical employee could see once access was granted [2][7].

The mechanisms in the current journal system might restrain malicious intend. However it does not guarantee prevention of unintentional access or disclosure of information[12]. What is missing is the enforcement of secure information flow policies. Unintentional access or disclosure of information can be prevented by enforcing policies that define secure information flow.

The prototype will model a simplified scenario of hospitals with different actors accessing a patient journal. The bulk of information on the journal system is extracted from newspaper articles hence there is a high uncertainty of how the system really works. Therefore many assumptions are made about the current system when programming the prototype.

1.4.2 Scope

The objective of the project is to do a secure implementation of the prototype described above. Secure exchange of patient journal is ensured using Matrix and the endpoints are secured using IFC.

A successful project is one that fulfills these criteria:

- Evaluation of Matrix security model
- Survey of IFC tools and selection of tool.
- Implement a prototype distributed system running on Matrix, using the chosen tools
- Demonstrate increased security guarantee with Matrix and IFC

1.4.3 Why Matrix?

In the Digital Strategy 2016-2020 the Danish Agency of Digitisation defines initiative 7.2 as *"Common standards for secure exchange of information"*. The large number of software systems in the Danish public sector has created a need for an uniform way of exchanging data across different application in a secure manner[20].

The initiative has similarities to the issue Matrix is trying to solve with fragmented IP communication. With Matrix security guarantees and their long term goal as a generic HTTP messaging protocol there is a strong case for using Matrix as a communication channel in this case study.

1.5 Method

1.6 Threat model

The threat model is defined in the context of confidentiality and integrity.

- The adversary has the ability to observe information sent over the network.
- The adversary can generate input to the system .
- The adversary can observe public output.

1.7 Contribution

The contributions to the field are the findings of secure implementation using Paragon and how they compare to similar findings from secure implementation with JIF.

The thesis also contributes with the interface created between Paragon and Matrix making it possible to develop other secure applications on top of secure communication channel Matrix provides.

1.8 Structure of thesis

Chapter 2 sets the foundation for the thesis and introduces relevant information and background. Chapters 3 analyzes the Matrix security model and survey IFC tools. Chapter **??** goes in depth with design of the solution. The results are then presented and discussed in Chapter **??**. The thesis is wrapped up in the conclusion section Chapter 6.

1.9 Summary

2 Background

This chapter builds the foundation for the thesis introducing relevant concepts and disciplines for the thesis.

Section 3.1 provides an evaluation of the Matrix security model. The security model is evaluated in the context of a secure messaging system. The paper *SoK: Secure Messaging* describes a evaluation framework for evaluating secure messaging systems. They define several security properties related to such systems [21] which will be presented.

All secure messaging systems are based on the Double Ratchet algorithm which will also be described.

Finally the architecture of Matrix and concepts related to Information-Flow Control will be introduced.

2.1 Information security

Information security is the discipline of protecting information. The key principles in information security are expressed through the CIA model. For a system to be secure these principles should be guaranteed [14].

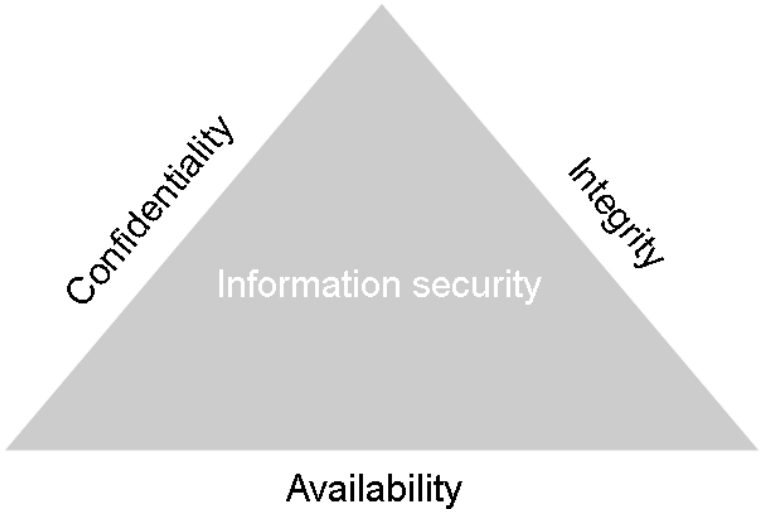

Figure 2.1: CIA triad

Confidentiality Confidentiality is keeping information secret from unauthorized people. This is a major goal in information security. Encryption and access control are common ways of ensuring confidentiality [14].

In a secure messaging system confidentiality would be guaranteed if the message being sent is only readable by the recipient and no one else [21].

Integrity Integrity is providing that information is unaltered and can only be changed by authorized people. If information is intercepted and changed during transit it would be a violation of integrity [14]. More specifically for a secure messaging it would mean that no altered message is accepted by the recipient [21].

Availability Making sure that information is accessible to authorized people is the goal of availability. Denial of Service attack [1] are common attacks targeting availability.

Availability is generally more related to the system being available where the information itself plays a minor role [14].

Depending on the type of system other properties must be satisfied as well.

2.1.1 Security properties

The goal in a secure messaging system is to protect the messages being sent. The following properties are related to protecting messages.

Authentication When a message is received the participant can verify that the message was sent from the actual sender. Furthermore a participant will receive evidence from a participant in a conversation that hey hold a known long-term secret.

Perfect Forward Secrecy If all keys are compromised than the decryption of any previously sent message should not be possible. Hence all previous messages would be secure however all future messages would be insecure

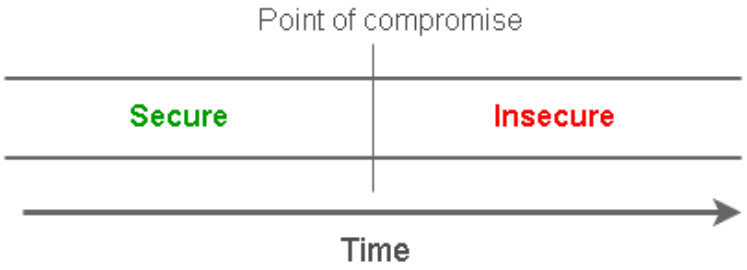

Figure 2.2: Forward secrecy

Backward secrecy If all keys are compromised than the decryption of *future* messages should be possible. This property also goes by the names *future secrecy* and *post compromise security*.

[1] https://en.wikipedia.org/wiki/Denial-of-service_attack

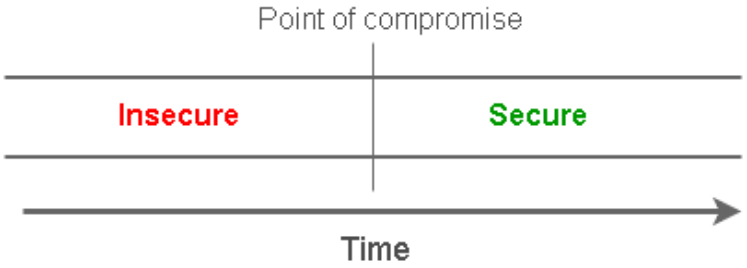

Figure 2.3: Backward secrecy

2.1.1.1 Other security properties

Participant Consistency Whenever a message is accepted by a participant all participants are guaranteed to have identical view of the participant list.

Destination Validation When a participant receives a message it can be verified that the participant was the intended recipient.

Anonymity Preserving The anonymity of the participants should be preserved and not linking any key identifiers.

Speaker Consistency There is consensus among the participants on the sequence of messages they receive by each participant. There might be a mechanism for checking consistency whenever a message is sent or after it has been received.

Causality Preserving Messages must not be displayed before the message that originally precedes it has been displayed.

Global Transcript A global order where all messages are viewed in the same order for all participants.

Deniability Deniability is a property where other participants cannot confirm that the message being sent was from the sender. Yet during the conversation there will be assurance for the recipient that the message being sent was authentic and sent by the sender [21].

- *Message Unlinkability:* A deniability property that gives no guarantees that if a participant sent a message that other messages was sent by that participant as well.
- *Message Repudiation:* It can not be proved that a message was authored by a participant given the conversation transcript and all cryptographic key material.
- *Participant Repudiation:* It can not be proved that a participant was in a group conversation without his conversation transcript and cryptographic key material.

The following properties are also defined in the paper *SoK: Secure Messaging* but are less relevant for security.

Group

- *Computational Equality:* The computational load is equal for all participants.
- *Trust Equality:* There is equal trust among all participants.
- *Subgroup messaging:* In the same conversation a participant can send messages to a subset of the participants.
- *Contractible Membership:* When a participant leaves a conversation the protocol does not need to restart.
- *Expandable Membership:* When a participant joins a conversation the protocol does not need to restart.

Adoption

- *Out-of-Order Resilient:* Messages received out-of-order should be accessible when received.
- *Dropped Message Resilient:* On a unreliable network messages might be dropped in transit however it should not prevent decryption of future messages.
- *Asynchronous:* Messages can be sent securely to recipients while they are offline.
- *Multi-Device Support:* A participant can have multiple devices in a conversation and each device must be synchronized and should have the same historical conversation view
- *No Additional Service:* There is no requirement of additional infrastructure being setup other than the participants.

2.1.2 Concepts

In information security standard security properties are achieved using cryptography.

2.1.2.1 Deffie-Hellman Key Exchange

2.1.2.2 Key Deriviation function

2.1.2.3 Unknown key share attack

2.1.2.4 Double Ratchet algorithm

The Double Ratchet algorithm has three stages. Before the Double Ratchet algorithm can be used the two parties communicating need to agree on a shared secrey key. In the Signal protocol this is achieved with Triple Deffie-Hellman protocol *(TripleDH)*.

Triple Deffie-Hellman protocol

The Triple Deffie-Hellman protocol is a *key agreement protocol*. It involves a server and two parties; Alice and Bob.

The TripleDH protocol is characterized by three phases:

1. *Publishing keys:* A identity key and several prekeys belonging to Bob is published by him to a server.
2. *Sending initial message:* Alice sends an initial message to Bob. A prekey bundle is obtained by Alice from the server in order to send an initial message to Bob.
3. *Receiving initial message:* Alice's message is received and processed by Bob.

Publishing keys Bob needs to register a *prekey bundle* to the server if he wants Alice to be able to send him messages. Alice will likewise have registered a prekey bundle so anyone can to anyone wants to start a message conversation with her. The prekey bundle exists of:

- Identity key IK_B. This key is only published once by Bob.
- Signed prekey SPK_B. This key is reuploaded again after some period of time (eg. after each week or each month).
- Prekey signature $Sig(IK_B, Encode(SPK_B))$. This key is also reuploaded again like the signed prekey.
- Set of one-time prekeys $(OPK_B{}^1, OPK_B{}^2, OPK_B{}^3, ...)$. These keys are uploaded by Bob occasionally. Bob is informed by the server when there are few one-time prekeys left.

  To ensure forward secrecy the private key of the one-time prekeys are deleted once Bob received messages that uses them. The signed prekey is deleted as well. However Bob might hold on to it for some time to get the messages that was delayed.

Sending initial message Alice retrieves Bobs public keys from the server. She receives one of Bob's single one-time prekey. The server deletes the one-time prekey that was send.

Alice verifies the prekey signature if the verification fails the protocol is aborted.

Alice provides the following public keys to generate a shared secret:

- Identity key IK_B. Her own identity key.
- Ephemeral key EK_B. The public key from a generated ephemeral key pair.

Now she can generate the shared secret.

The calculation and concatenation of DH4 is optional since there might not be anymore one-time prekeys on the server.

Alice's initial ciphertext is typically used as the first message in a post- X3DH communication protocol, such as the Double Ratchet protocol from section 3.1. The ciphertext has two roles, serving as the first message within some post-X3DH protocol, and as part of Alice's X3DH initial message to Bob

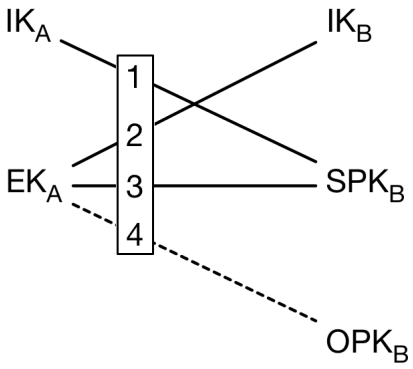

Figure 2.4: Calculations of DH1-DH4 [13].

Receiving initial message Phase three of the X3DH is mostly the same as phase two. Bob retrieves Alice's initial message which contains Alice's identity key and ephemeral key from the message. Bob will then load his identity private key and the private key(s) corresponding to the signed prekey and one-time prekey that Alice used. Bob repeats the same steps with DH and KDF calculations to derive his own SK and then deletes the DH values the same as Alice did. Afterwards, he constructs the AD byte sequence, and in the end, tries to decrypt the initial ciphertext using the SK and AD. The decryption is the only difference Bob does to what Alice did on her side. If the decryption fails, Bob will delete the SK and the protocol aborts and the participants need to restart the protocol from the start. If the decryption is successful, he gets the information that Alice had encrypted, and the protocol is complete for Bob. He deletes any onetime prekey private key that was used during the protocol, to uphold the forward secrecy and not get it compromised.

KDF chain

KDF Chain is a core concept in the Double Ratchet algorithm. The job of the KDF chain throughout the Double Ratchet session is to store each parties KDF key for three chains: a root chain, a sending chain, and a receiving chain. The KDF chains are both part of the Diffie-Hellman ratchet step and the Symmetric-key ratchet step. While the parties exchange messages, they need to exchange new Diffie-Hellman public keys. The secrets from the DH output become the inputs to the root chain for the KDF chain, and then the output keys from the root chain become new KDF keys for the sending and receiving chains. This is the Diffie-Hellman ratchet. The outputs from the Diffie-Hellman ratchet, the sending and receiving chains advances for each sending and receiving message, by using the chains as inputs in the KDF and the output keys are then used to encrypt and decrypt messages. This is called the symmetric-key ratchet.

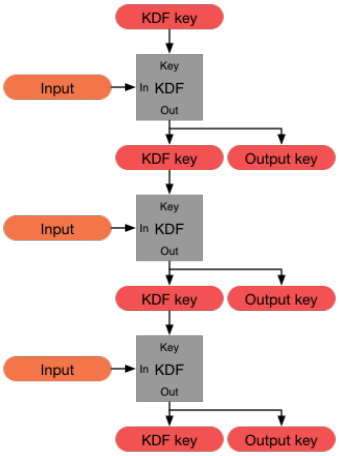

Figure 2.5: Processing of three inputs and the resulting outputs in KDF chain [17].

Symmetric ratchet

The symmetric-key ratchet uses KDF chains for its sending and receiving chains. The output keys are unique message keys which are used to either encrypt or decrypt messages. The KDF keys for the symmetric-key ratchet chains will be called chain keys throughout the rest of the Double Ratchet description. The KDF chains used for the sending and receiving chains are constant; they do not need to be random or a secret because the chain key is derived from the Diffie-Hellman KDF chain which is cryptographically secure. The documentation explains the constant can be a single byte 0x01 for the message key and then a single byte 0x02 for the chain key. The sending and receiving chains ensure that each message is encrypted or decrypted with a unique key and can be deleted after use. There is only a single symmetric-key ratchet step to calculate a new message and chain key from an already given chain key. The diagram below shows two steps in this process. The first KDF gets a chain key from the Diffie-Hellman KDF chain and outputs a new chain key for the next KDF and a message key to encrypt or decrypt a message.

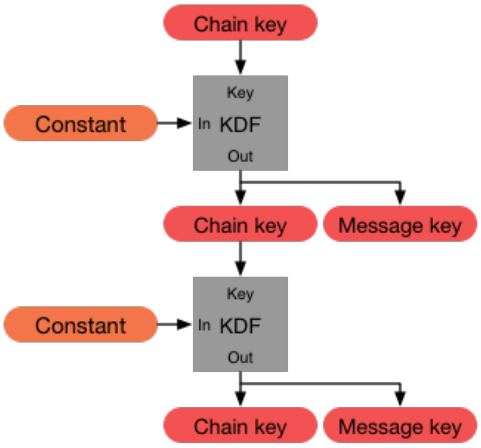

Figure 2.6: Symmetric key ratchet [17].

The message key that is derived from the KDF chain is not used later on to derive new message keys or chaining keys. Because of this, it is fully possible to store the message key without affecting the security of other keys, only the message that belongs to the particular message key. It is quite useful when the protocols handles out-of-order messages because a participant can store the message key and decrypt the message later when they receive the correct message for that message key.

Deffie-Hellman ratchet

The Double Ratchet is formed by combining the symmetric-key ratchet and the Diffie-Hellman ratchet. If the Double Ratchet did not use the Diffie- Hellman ratchet to compute new chain keys for the sending and receiving chain keys, an attacker could steal one of the chain keys and then compute all future message keys and decrypt all future messages. For this to work, each party generates a DH key pair, a public and a private key, which will be their first ratchet key pair. When a message is sent, the header must contain the current public key. When a message is received, the receiver checks the public keys that are given with the message and do a DH ratchet step to replace the receiver's current ratchet key pair with a new one. The result is a kind of "ping-pong" behavior as the two parties take turns replacing their key pairs. The attacker will have a harder time to get any valuable information from the parties, since if one of the messages gets compromised, and the attacker learns the value of the current private key, it will not make any difference since the private key will soon be replaced with a new, uncompromised key. From this point onwards, this section will present an example of how the Diffie-Hellman ratchet works to get a shared sequence of DH outputs. The initialization starts with Bob sending his ratchet public key to Alice, while Bob does not know Alice's ratchet public key. Alice will now do the initialization step by performing a DH calculation between her ratchet public key and Bobs ratchet public key. Figure

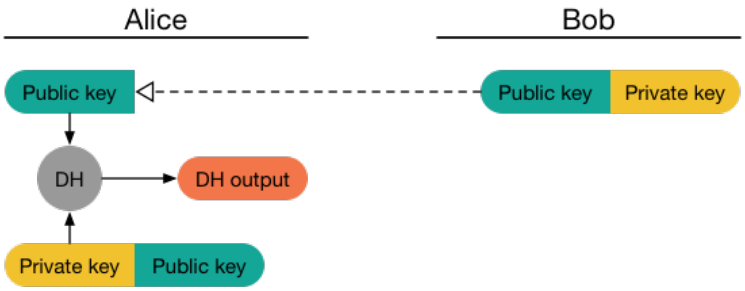

Figure 2.7: Diffie-Hellman ratchet 1 [17].

The figure shows Alice advertise her ratchet public key to Bob after she is done with her Diffie-Hellman calculation. When Bob receives Alice's initial message, he performs a Diffie-Hellman ratchet step by calculating the new DH output between Alice's ratchet public key and his ratchet private key, which equals Alice's DH output. The DH outputs are equal because the figures are a simplification of the DH ratchet, and there is a KDF chain which uses a Root Key (shared secret between Alice and Bob) to output the same keys. He then replaces his ratchet key pair and calculates a new DH output.

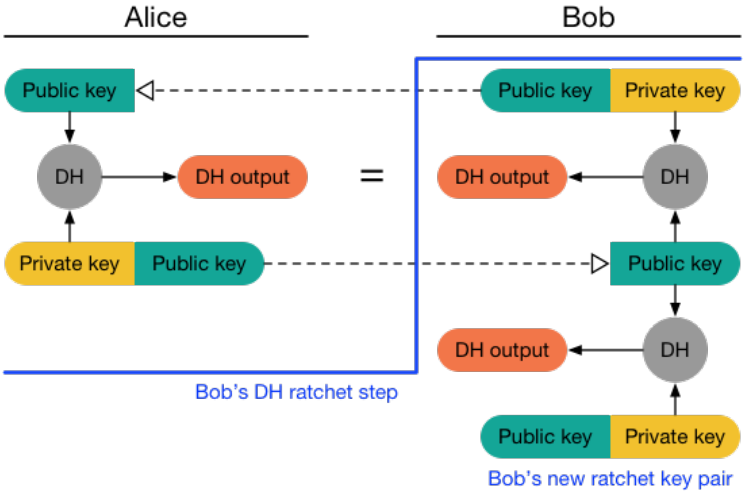

Figure 2.8: Diffie-Hellman ratchet 2 [17].

Bob sends his next message with the new ratchet public key. Eventually, Alice receives Bob's new message with his new ratchet public key. Alice derives a newDHoutput with her ratchet private key and Bob's new public key to get the same DH output as Bob for decrypting the message and then generates a new DH

ratchet key pair to replace her old key pair. With the new DH ratchet key pair, Alice derives another DH output with her new ratchet private key and the same ratchet public key from Bob, which is used in the next message exchange. These exchanges continue for each message sent with a new ratchet public key and new DH output.

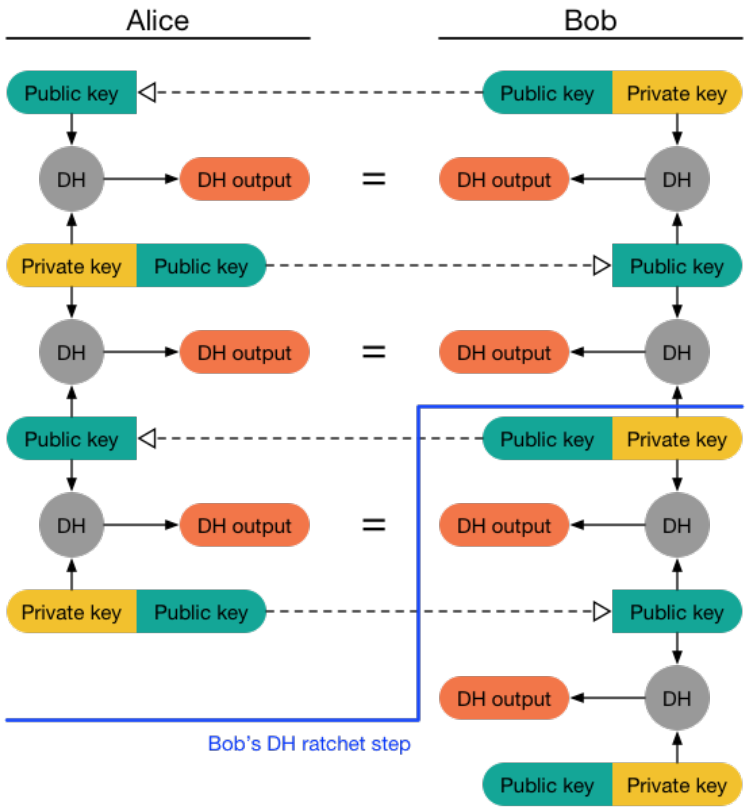

Figure 2.9: Diffie-Hellman ratchet 4 [17].

Until now the DH outputs have only been called outputs to simplify the description of the DH ratchet, but the DH outputs are used in a KDF to get the sending and receiving chain keys for the symmetric-key ratchet. Figure 3.5 shows the DH outputs are changed with sending or receiving chains. The first time Alice sends out her message with her ratchet public key and DH output, a sending chain is derived from the DH output through a KDF chain, and Bob on his side derives a receiving chain key from hisDHoutput by using Alice's public key and his private key. Bob then derives a new sending chain key from his second DH output.

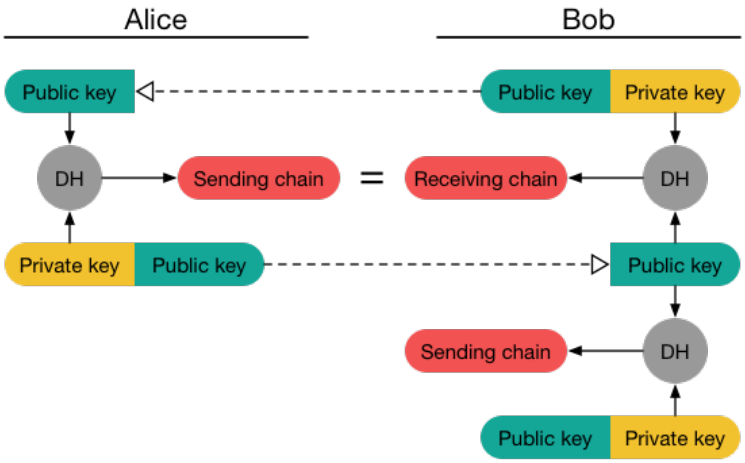

Figure 2.10: Diffie-Hellman ratchet 5 [17].

The figures shown until now have been a simplification of the DH ratchet. The DH ratchet takes the output keys and uses them as KDF inputs to a root chain, and then the KDF outputs are used as a sending or receiving key. Using a KDF chain here improves the resilience and break-in recovery. Figure 3.6 shows that the DH ratchet updates the root KDF chain twice, and uses the KDF output keys as new sending and receiving chain keys

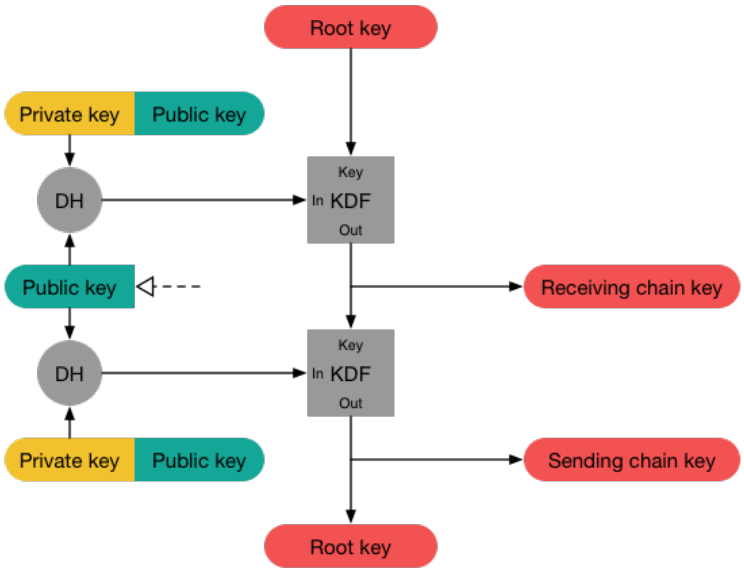

Figure 2.11: Diffie-Hellman ratchet 7 [17].

Double ratchet

Double Ratchet is the algorithm we get by combining symmetric-key ratchet and Diffie-Hellman ratchet. When a message is sent or received, a symmetric-key ratchet step is applied to the sending or receiving chain to derive the message key. When a new ratchet public key is received, a DH ratchet step is performed prior to the symmetric-key ratchet to replace the chain keys.

This section is about how these algorithms work together to form the Double Ratchet, seen from Alice's perspective. The figure shows after Alice has done her initialization with Bob's ratchet public key, and the Root Key (RK) is the shared secret used as the initial root key for the KDF chain. Alice generates her ratchet key pair and then sends the DH output to the root KDF chain to calculate a new root key and a sending chain key (CK). The figure is split up into four different parts. The ratchet is where the ratchet key pair changes and sends Alice's ratchet private key and Bob's ratchet public key to generate a DH output for the root KDF chain input. The sending column is where the message key for encryption generates through the symmetric-key KDF chain. The last column is where the message key for decrypting the received message gets also derived by the symmetric-key KDF chain. To ensure that the secrecy throughout the Double Ratchet is upheld, the old RK gets deleted after it has been used to derive a new RK.

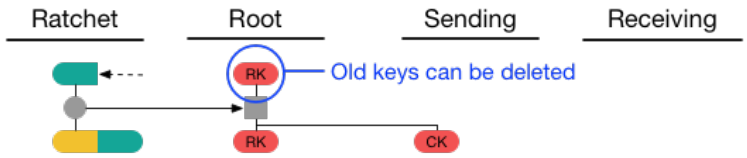

Figure 2.12: Double ratchet 1 [17].

The next figure shows Alice sending her first message to Bob, message A1. The sending CK is used on a symmetric-key ratchet step to derive a new CK and a message key, A1, to encrypt her message. The new CK is stored for later use, while the old CK and the message key can be deleted since it no longer is of any use to Alice.

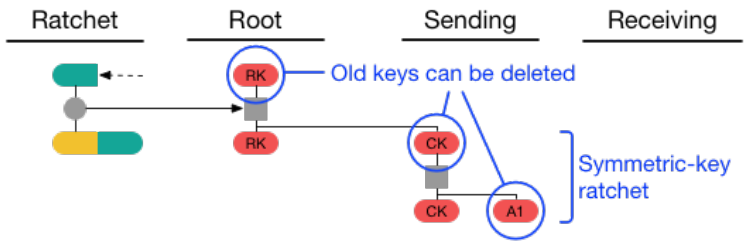

Figure 2.13: Double ratchet 2 [17].

The figure shows Alice receiving her first response from Bob, message B1, and he has sent his new ratchet public key, which means Alice needs to calculate a new ratchet key pair. Alice applies a new DH ratchet step to derive a new receiving and sending chain keys. Alice uses her old ratchet private key and Bob's new public key, to derive a new RK and a CK for the receiving KDF chain. The receiving chain key is used to derive a new receiving chain key and a message key to decrypt Bob's message. Then she derives a new DH output for the next root KDF chain with her new ratchet private key to derive a new RK and a sending CK.

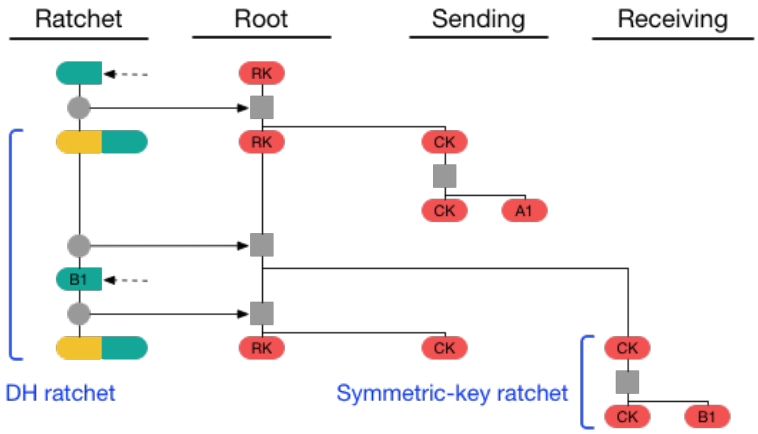

Figure 2.14: Double ratchet 3 [17].

The next figure shows how many ratchet steps Alice does when sending a message A2, receiving a message B2 with Bob's old ratchet public key, and then send two new messages A3 and A4 to Bob. Alice received a message B2 with Bob's old ratchet keys, which means that Alice only needs to do a symmetric-key ratchet step to derive a new receiving CK and a message key to decrypt message B2. Before Alice sends her second message A2, she needs to do a symmetric-key ratchet step to derive a new sending CK and a message key to encrypt her message A2. The same must be done with message A3 and A4, by ratcheting the symmetric-key ratchet two more times to derive the correct message keys.

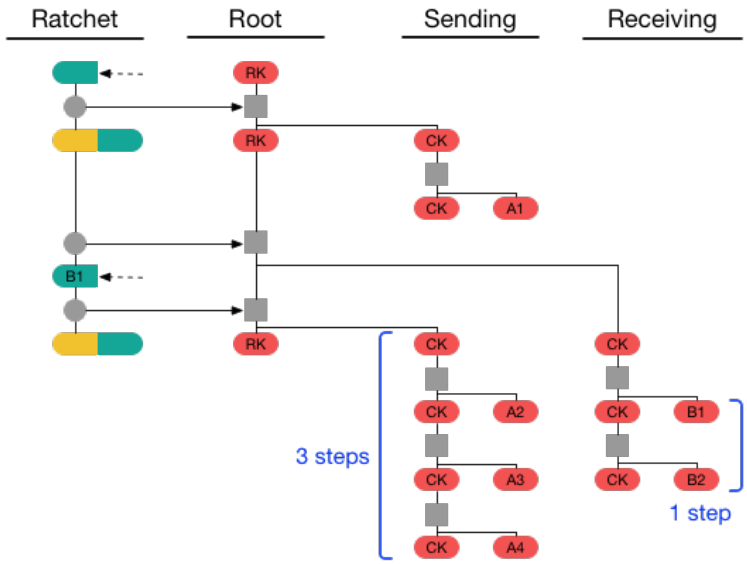

Figure 2.15: Double ratchet 4 [17].

This figure shows the state when Alice receives messages B3 and B4 from Bob with his new ratchet public key, and the sending of Alice's last message, A5. Alice receives new messages from Bob with his new ratchet public key, and she first derives a new DH output to ratchet the root KDF chain to get a new RK and a new receiving CK to decrypt Bob's messages. The receiving CK is used to run the symmetric-key ratchet step two times, ones to derive a new CK and a new message key for decryption of message B3 and then another ratchet step to derive the second CK and message key for message B4. Alice generates a new ratchet key pair and uses her new ratchet private key to derive a new RK and a new sending CK with Bob's new ratchet public key. The sending CK is used to do a symmetric-key ratchet step to derive a new CK and a message key to encrypt her new message, A5.

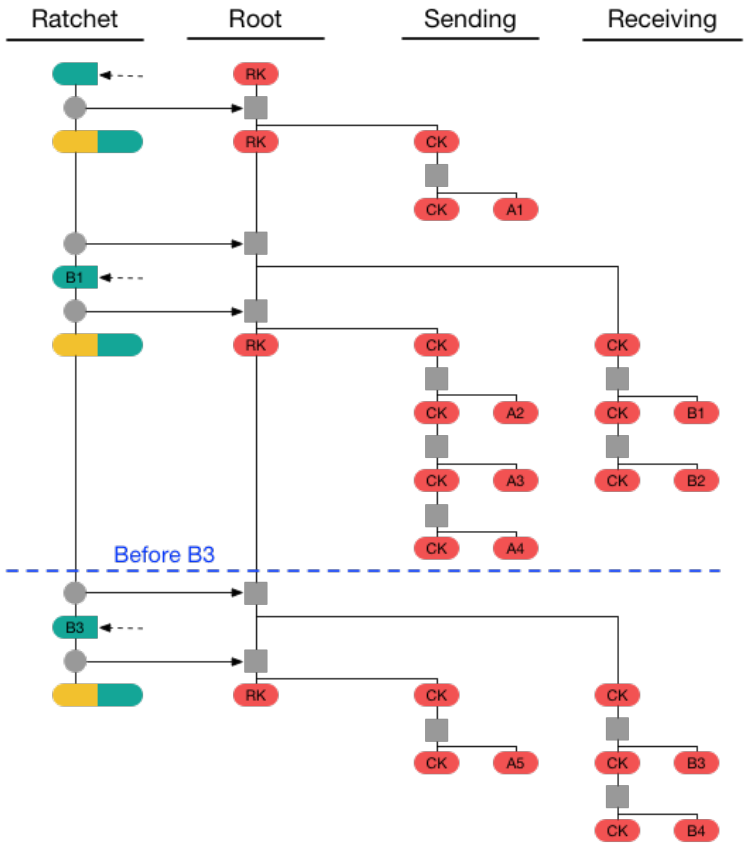

Figure 2.16: Double ratchet 5 [17].

2.2 Matrix

2.2.1 Goals

2.2.1.1 Short term goal

2.2.1.2 Long term goal

2.2.2 How does it work?

2.2.3 Architecture

2.2.4 Matrix specification

2.2.5 End-to-end Encryption

2.3 Information Flow Control

2.3.1 The Lattice Model

adsadsad

2.3.2 Noninterference

dasdsa

2.3.3 Static policies

sadsad

2.3.4 Dynamic policies

sadsadsad

2.3.5 Declassification

Taking some specific information and changing it to a lower security classification.

Identify: What to classify, who declassifies, where the declassification happens and when the declassification happens

2.4 Summary

3 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* [21] and *The Olm Cryptographic Review* by NCC Group [15].

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.

3.1 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*

3.1.1 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 [21].
This evaluation will inherit the described threat model.

3.1.2 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) [21].

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 3.1: Evaluation of Signal (TextSecure) [21].

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 [21].

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 [21].

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 [21].
- ***Destination validation***. The Deffie-Hellman ratchet provides this property since the recipients public key is used to generate the chain key [21].

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 [11].

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.

3.1.3 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 [21].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 3.2 illustrates the setup with three participants.

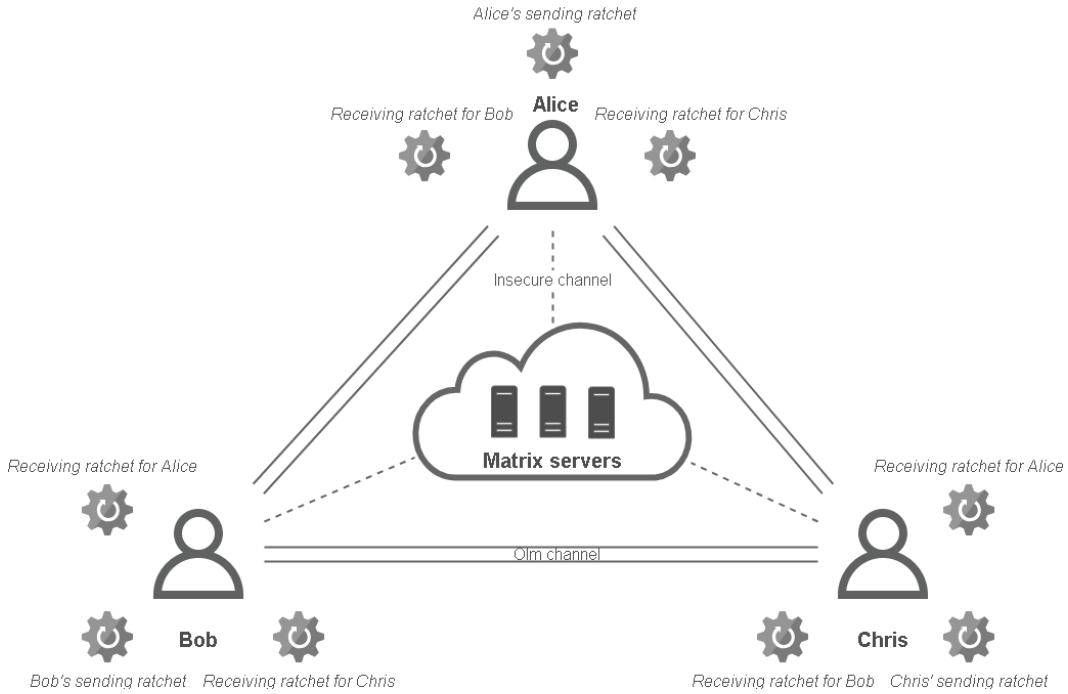

Figure 3.2: Conceptual model of Megolm with three participants.

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.

3.1.3.1 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 [15] [**?**].

- *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 [15] [**?**]. This would scale poorly for a large group or when sending large-sized messages.

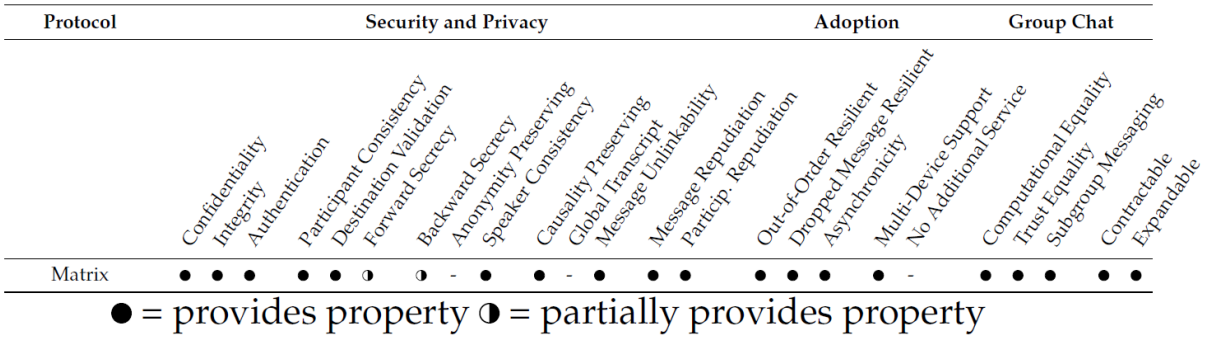

Figure 3.3: Evaluation of Matrix Security.

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 [15].

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 [15].

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*** [15].

Other properties

The multi-cast encryption design does not provide *participant consistency* [21].

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 [15].

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.

3.1.4 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.

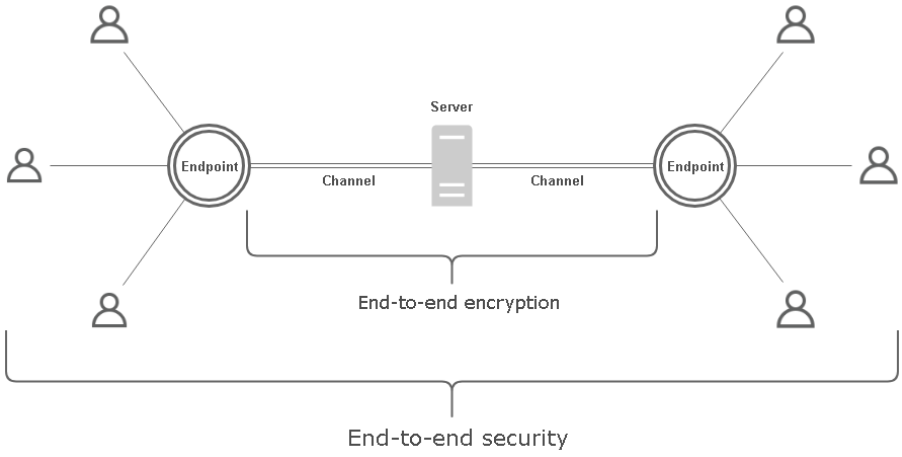

Figure 3.4: End-to-end security.

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 3.5 below.

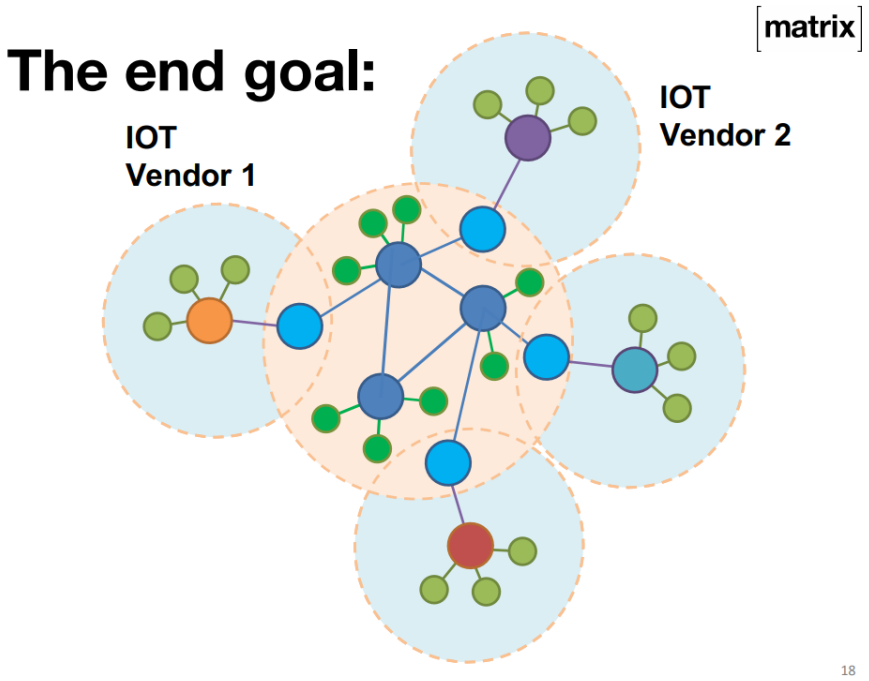

Figure 3.5: End goal for Matrix IoT

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.

3.1.5 Summary

In this section an evaluation of Matrix security was presented. Several security properties are a part of Matrix security model with forward secrecy and backward secrecy being provided depending on the Matrix configuration.

End-to-end encryption is not the end of security. Other security measures must be taken to provide confidentiality and integrity. Information Flow Control is such measure and the next section will present a survey on Information Flow Control.

Bibliography

[1] Op imod 90.000 ansatte kan kigge i din journal - Indland. URL https://jyllands-posten.dk/indland/ECE6715461/op-imod-90000-ansatte-kan-kigge-i-din-journal/.

[2] Adgang til sundhedsdata - sundhed.dk. URL https://www.sundhed.dk/borger/service/om-sundheddk/om-portalen/datasikkerhed/andres-dataadgang/adgang-til-sundhedsdata/.

[3] CWE - CWE-359: Exposure of Private Information ('Privacy Violation') (3.2). URL https://cwe.mitre.org/data/definitions/359.html.

[4] Journal fra sygehus. URL https://www.sundhed.dk/borger/min-side/min-sundhedsjournal/journal-fra-sygehus/.

[5] Principles for a More Informed Exceptional Access Debate - Lawfare. URL https://www.lawfareblog.com/principles-more-informed-exceptional-access-debate.

[6] Google Plus Will Be Shut Down After User Information Was Exposed - The New York Times. URL https://www.nytimes.com/2018/10/08/technology/google-plus-security-disclosure.html.

[7] Kontrol af opslag - sundhed.dk. URL https://www.sundhed.dk/borger/service/om-sundheddk/om-portalen/datasikkerhed/portalens-beskyttelse-af-data/kontrol-af-opslag-e-journal/.

[8] FAQ | Matrix.org. URL https://matrix.org/docs/guides/faq.

[9] 91,000 state Medicaid clients warned of data breach | The Seattle Times. URL https://www.seattletimes.com/seattle-news/health/91000-state-medicaid-clients-warned-of-data-breach/.

[10] Matt Bishop. *Introduction to computer security*. Addison-Wesley, Boston, 2005. ISBN 978-0321247445.

[11] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A Formal Security Analysis of the Signal Messaging Protocol. *Proceedings - 2nd IEEE European Symposium on Security and Privacy, EuroS and P 2017*, (November):451–466, 2017. ISSN 13484214. doi: 10.1109/EuroSP.2017.27.

[12] Laurinda B. Harman, Cathy A. Flite, and Kesa Bond. Electronic Health Records: Privacy, Confidentiality, and Security. *Virtual Mentor*, 14(9):712–719,

sep 2012. ISSN 1937-7010. doi: 10.1001/virtualmentor.2012.14.9.stas1-1209. URL `http://virtualmentor.ama-assn.org/2012/09/stas1-1209.html`.

[13] Moxie Marlinspike and Trevor Perrin. The X3DH Key Agreement Protocol. Technical report, 2016. URL `https://signal.org/docs/specifications/x3dh/x3dh.pdf`.

[14] Katina Michael. *The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice*, volume 31. 2012. ISBN 9781597496537. doi: 10.1016/j.cose.2012.03.005. URL `http://linkinghub.elsevier.com/retrieve/pii/S0167404812000557`.

[15] NCC Group. Olm Cryptographic Review. (November):1–27, 2016. ISSN 0737-4038.

[16] P. D. Pacey and J. H. Purnell. OWASP Top 10 - 2017. *International Journal of Chemical Kinetics*, 4(6):657–666, 1972. ISSN 10974601. doi: 10.1002/kin.550040606.

[17] Trevor Perrin and Moxie Marlinspike. The Double Ratchet Algorithm. Technical report, 2016. URL `https://signal.org/docs/specifications/doubleratchet/doubleratchet.pdf`.

[18] Fiza Abdul Rahim, Zuraini Ismail, and Ganthan Narayana Samy. Information privacy concerns in electronic healthcare records: A systematic literature review. In *2013 International Conference on Research and Innovation in Information Systems (ICRIIS)*, pages 504–509. IEEE, nov 2013. ISBN 978-1-4799-2487-5. doi: 10.1109/ICRIIS.2013.6716760. URL `http://ieeexplore.ieee.org/document/6716760/`.

[19] Andrei Sabelfeld and Andrew C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, 2003. ISSN 07338716. doi: 10.1109/JSAC.2002.806121.

[20] The Government, Local Government Denmark, and Danish Regions. *The Digital Strategy - A stronger and more secure digital Denmark*. Agency for Digitisation, 2016. ISBN 9789400769250 | 9400769245 | 9789400769243. doi: 10.1007/978-94-007-6925-0_9. URL `https://en.digst.dk/media/14143/ds{_}singlepage{_}uk{_}web.pdf`.

[21] Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith. SoK: Secure Messaging. pages 232–249, 2015. doi: 10.1109/SP.2015.22.