Teaching Activity “Klemmchat”

Luisa Gebhardt, Marc Leinweber
Institute of Information Security and Dependability (KASTEL)
Karlsruhe Institute of Technology (KIT)
luisa.gebhardt9@kit.edu, marc.leinweber@kit.edu

This document presents the teaching activity as it was performed as part of the study for the paper “Grasping the Concept of Decentralized Systems for Instant Messaging “[1].

The aim of the activity is to teach students the advantages and disadvantages of different centralized or decentralized algorithms in instant messengers. The focus is on the reliability and power structure in the instant messengers and their influence on the usage of the instant messenger.

Klemmchat simplifies the complicated algorithms of such instant messengers to convey them to students without computers and technical knowledge. In doing so, Klemmchat neglects the confidentiality of the messages and does not use any encryption mechanisms.

Klemmchat simulates a group chat, as it exists in modern messengers, with the help of Duplo bricks, which are exchanged between groups of students according to rules depending on the respective simulated algorithm. After a short explanation, the students' task is to manipulate the respective algorithm to harm the messenger; the students thus play attacker and honest provider and user at the same time.

Klemmchat can be used and accompanied flexibly in the classroom. This document presents an exemplary lesson plan for the use of Klemmchat. The following learning objectives are to be achieved by the students:

The students are able to…

1. … describe the basic functionality of the different algorithms (centralized, distributed, decentralized) for instant messaging
2. … discuss the trade-offs of the different algorithms
3. … contrast which algorithm to use in a given scenario based on their characteristics
4. … evaluate instant messaging services, based on their underlying algorithm, in different usage scenarios

Klemmchat Structure

In most cases, communication in a messenger takes place from one user's client to another user's client. The messenger provider coordinates the message exchange and keeps certain parts of the service available.

The users' clients are the applications on the devices. The message m from a sender c_1 to recipient c_2 is thus sent from the app on the smartphone to the server of a provider. This provider stores the message in the message history between participants c_1 and c_2. This message history is then regularly retrieved from the provider by both participants for reading. Message writing and retrieval can be organized architecturally and politically in different ways. In Klemmchat, a centralized, a distributed and a decentralized variant are simulated.

[1] https://publikationen.bibliothek.kit.edu/1000150316

Klemmchat needs the following components to simulate the different algorithms:

Participants: A group of three to four students simulates a participant. We do not distinguish between the user and the application on the device. The students embody both the user and the application.

Message: A message consists of one or more Duplo bricks with the same sender and message text. The message text is written on the brick using a whiteboard marker.

Message Sender: The sender of a message is indicated by the color of the message brick. To make this assignment unique, each group receives bricks in a different color from the other groups.

Message Recipient: Klemmchat simulates a group chat. A message is always addressed to all participants. Explicit identification of a recipient is not necessary.

Provider: Depending on the respective algorithm, the provider is simulated by other groups/persons. More details are specified in the paragraph about the respective algorithm.

Message History: The message history is created by appending a received message brick to the bottom of the previously received message bricks. This creates a tower of message bricks. If you want to read the content of a message history, you read it from top to bottom by saying "*Color of sender* writes *message*" for each brick.

Write and Read Functions: These functions depend on the respective algorithm. To order the flow of the algorithms, there are read and write phases. In a write phase only writing is allowed, in the read phase the message history is read once.

Centralized Algorithm

In the central algorithm, the message history is stored on the server of a provider. This server is played by the teacher. So, there is only one version of the message history. If this message history is lost or the server or the teacher is not available or busy, it cannot be read or written.

If a group wants to write, the group labels a brick with the message and comes forward to signal that a message can be picked up. The teacher then walks to that group and checks to see if the color of the brick matches the color of the group. If the check is successful, the teacher picks up the brick and takes it to the desk where the message history is located and attaches the brick to the message history. If not, the teacher ignores the brick. To read, the teacher reads the message history out loud.

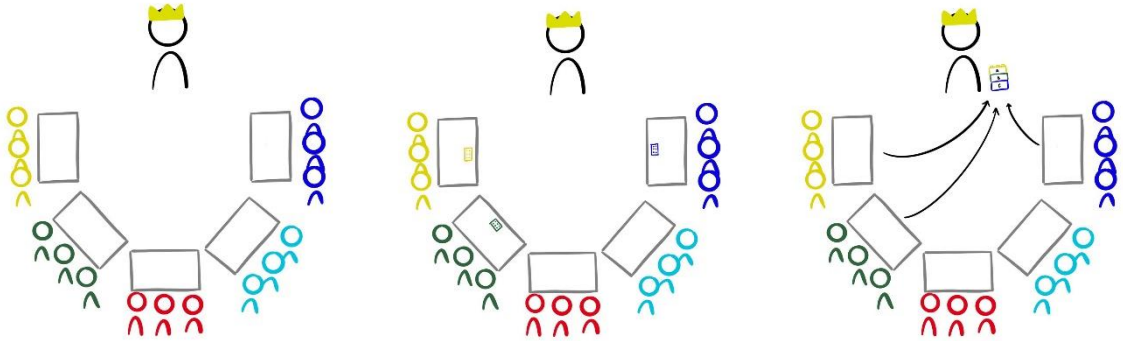

Figure 1: Illustration of the central writing process: The yellow, blue and green groups each write a message. These are collected by the teacher

The following aspects can be observed by the students:

Reliability:

- The teacher is overwhelmed with the number of messages to be processed simultaneously (susceptibility to DoS attacks).
- It may happen that by mistake the message history is no longer correct.

⇨ *The teacher represents a single point of failure with their message history. The instant messenger is not very reliable, as it often does not work.*

Power Structure:

- The teacher can intentionally ignore the messages of a group, even if the sender matches the color of the brick. They can therefore intentionally discriminate against participants structurally.
- The teacher can accept faulty messages.
- The teacher can censor or not accept swear words, but also messages that do not have harmful content. They can arbitrarily censor and moderate content.
- The teacher can also replace the contents of a message with others. In doing so, they can also pretend to be a participating group.

⇨ *The participants in the service must trust that the provider/teacher will act in the interests of the participants and not exploit their power over the messenger in a self-serving manner. This form of power distribution can be compared to a monarchy.*

Distributed Algorithm

In the distributed algorithm, the message history is stored distributed across multiple servers of a provider. This server group *r* is played by a selected group of students. Each student in group r has his or her own copy of the message history on the console. Thus, there are multiple versions of the message history. Ideally, these are always the same. The ability for coordination by a single provider is made possible by agreements within the manageable group of students.

If a participating group wants to write, the group labels one brick per student in group *r* with the same message and signals that a message can be picked up. One person in group *r* then walks to that group and checks that the color of the bricks matches the color of the group and that the bricks match each other. If the check is successful, the person picking up the bricks takes them and distributes them to all the other students in group *r*. They then take their brick to the desk where the message histories are located and attach the brick to their message history. Group *r* ignores the message if the check was not successful. Multiple messages may be accepted by group r at the same time. This may cause swaps in the message history. Simple swaps may be resolved manually. Nevertheless, it is important that students are aware of this problem.

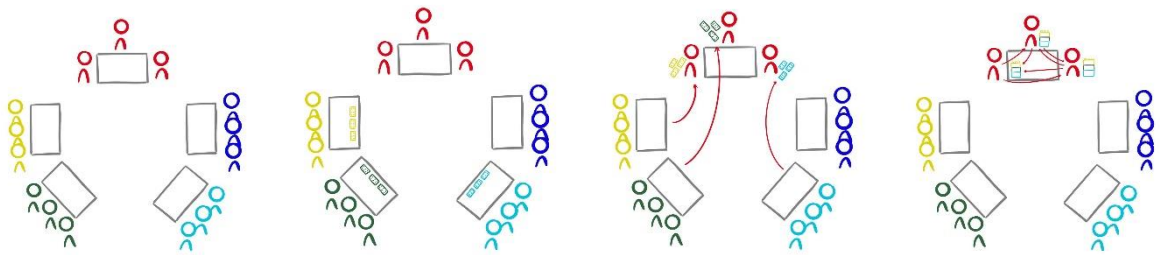

Figure 2: Illustration of the distributed writing process: The light blue, yellow and green groups each write a message. These are collected by one person in the red server group *r*. This person then distributes the bricks to everyone else in server group *r*.

For reading, each person in group *r* reads the message history. Then the students in group *r* agree on one of the message histories as the "correct" history. If there is disagreement, the opinion of the absolute majority counts. This message history is then used to respond.

The following aspects can be observed by the students:

Reliability:

- The group *r* can process several messages at the same time.
- If a message history is faulty, this can be corrected by the group *r*, since other copies of the history still exist.
- It is possible that there is no completely correct message history.

⇨ *The instant messenger is more reliable because there is no longer a single point of failure.*

Power Structure:

- The group *r* can collude and intentionally ignore the messages of a group, even if the color of the brick matches the sender and the messages match each other.
- The group *r* can collude and accept faulty messages.
- The group *r* can collude and censor or not accept swear words. But it is also possible to censor well-intentioned messages.
- The group *r* can collude and accept faulty messages.
- Group *r* can collude and substitute the contents of one message for another, impersonating another participating group.

⇨ *The participants in the service must trust that the provider/group r will act in the interests of the participants for the benefit of the messenger and will not exploit their power over the messenger for their own advantage. This form of power distribution can be compared to an oligarchy.*

Other:

- The algorithm is more elaborate and complicated.

Decentralized Algorithm

In the decentralized algorithm, each group plays its own server. Server and client coincide in this experiment. The message is sent to own servers hosted by the participants, which exchange the messages with all other servers.

The message history is stored in a distributed manner on the servers of all participants. Each group has its own copy of the message history on its table. So again, there are multiple versions of the message history. Ideally, these are always the same. Coordinating all the servers is more difficult because there is no longer a single provider. Collusion between each group to coordinate the read and write process is urgently needed if the goals of the messenger are to be met. Because of replication and collusion, a group cannot single-handedly manipulate the message history at will without the possibility of correction.

Now, since the bricks of one group can get into the hands of any other group, the color of the bricks alone is not enough to protect participants from identity theft: a message can be wiped off the brick and the brick can be reused. Therefore, additional digital signatures in the form of colored tape are used. Each group or brick color can be assigned the color of a specific adhesive tape. Instead of writing directly on the brick, a group writes its message on its own tape, which it has previously stuck on the brick. A message is now only valid if the color of the brick and the color of the tape match the identity

of the sender. The message cannot now be removed without leaving a residue, without invalidating the message, because the tape cannot be wiped off. The tape of one group must never be given to another group.

If a participating group wants to write, the group labels one brick per other group with the same message on tape and signals that a message can be picked up at their desk. One person from each other group then walks to the writing group and checks that the message is valid and that the bricks match. If the check is successful, each of these people takes the brick to their own table, where the group's message history is, and attaches the brick to their message history. Otherwise, the group ignores the message. Again, multiple messages can be accepted by the servers at the same time. This may cause swaps in the message history. Simple swaps may be resolved manually. Furthermore, it is important that the students recognize this problem and realize that it is much more difficult to keep the message histories consistent in this algorithm.

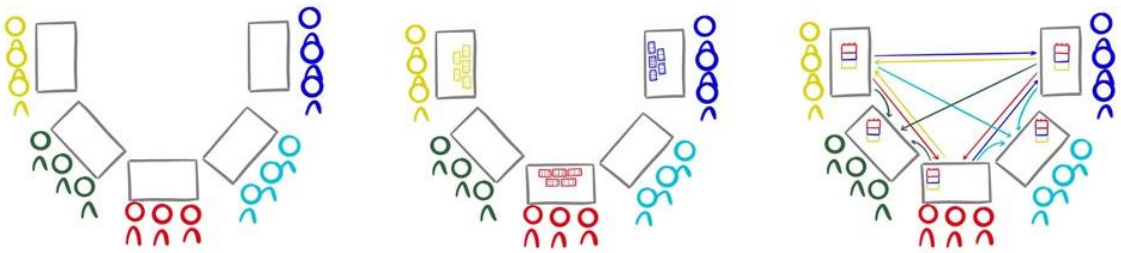

Figure 3: Illustration of the decentralized writing process: The yellow, red and blue groups each write a message. One person from each other group then picks up one or more messages from the other groups.

To read, a representative from each group reads their own message history. Then the representatives of each group agree on one of the message histories as the "correct" history. If there is disagreement, the opinion of the absolute majority counts. This message history is then used to respond.

The following aspects can be observed by the students:

Reliability:

- The servers can handle multiple messages at the same time.
- If a message history breaks, this can be corrected since other copies of the history still exist.
- It is possible that there is no completely correct message history.

⇨ *The instant messenger is more reliable because there is no longer a single point of failure. However, settlement can be more difficult and it is less likely that there will be an error-free process.*

Power Structure:

- The majority of groups must coordinate to accept faulty messages, reject correct messages, but also to censor swear words.
- Censorship or moderation of content is now only possible with difficulty, since majority decisions are always necessary. Due to the lack of moderation, individual participants can be discriminated against by other, few participants in chat messages.
- Due to the signatures, the message content cannot be changed afterwards.

⇨ *The participants in the service no longer have to trust that a provider will act in the interests of the participants, and the exploitation of power by individual providers is ruled out. The necessary trust is placed in the majority of a federation instead of in individuals. This form of power is comparable to a democracy.*

Other:

- Due to the coordination between the groups and the redundant work to be done by the groups, the algorithm is more elaborate and complicated than the centralized and the distributed variants.

Remark:

Instead of reading the message histories aloud, they can be held under a visualizer or document camera. It is only important that the students can perceive the message histories. In this case, it is also possible that something else is read during the reading out than is actually written in the history. Attacks that are not observed by students serve no purpose in Klemmchat.

Lesson plan for the Exemplary Embedding of Klemmchat

In the following table, an exemplary teaching structure is given, which teaches the algorithms described above and the associated learning objectives. It is intended for students in the 10th grade and above. In particular, it is helpful if the students already have knowledge about the distribution of power in political systems.

The time allotted is very tight; it is worthwhile to extend and give more space to the individual experiments and discussions.

Time	Phase	Content	Activity
00:00 5 min	Introduction	Welcome, objectives of the lesson, introductory question about the messengers used by the students.	Teacher-student discussion
00:05 10 min		Explanation of the basic structure of a messenger and terminal chat, note on the lack of encryption.	Lecture
00:15 5 min	**Experiment 1:** Introduction	Explanation of the centralized algorithm	Lecture
00:20 10 min	Development	Simulate the central algorithm in several rounds. Each round consists of a writing phase and a reading phase. After each round, all noticeable features are collected. In addition, the messenger can be reset (return bricks). In the first round, no manipulation attempts should be made yet. In the following rounds, features described above should be discovered exploratively. The teacher can provide support here.	Teamwork
00:30 5 min	Consolidation	Summary of striking features. The central position and trust in the provider as well as the connection to power distribution are to be emphasized.	Teacher-student discussion
00:35 5 min	**Experiment 2:** Introduction	Explanation of the distributed algorithm	Lecture
00:40 10 min	Development	Simulate the distributed algorithm in several rounds (see centralized algorithm).	Teamwork

00:50 10 min	Consolidation	Summary of the striking features. It is important that differences and similarities are identified in comparison to the previous algorithm.	Teacher-student discussion
01:00 5 min	**Experiment 3:** Introduction	Explanation of the decentralized algorithm	Lecture
01:05 10 min	Development	Simulate the decentralized algorithm in several rounds (see centralized algorithm).	Teamwork
01:15 5 min	Consolidation	Summary of striking features. It is important to identify differences and similarities by comparing them to the two previous algorithms.	Teacher-student discussion
01:20 10 min	Conclusion	Reflection questions: • Which algorithm would the students choose for their own messenger? Why? • What algorithm do the messengers use that were initially named by the students? • Does what is learned influence the students' usage decisions?	Teacher-student discussion