1. Introduction. (Simon)
2. Planning. (Matze)
   1. Goal.(Matze)
   2. Core idea. (Matze)
3. Concept.(Simon)
   1. Requirements analysis. (Simon) – Not needed
   2. Technical frameworks. (Simon)
   3. Technical concept. (Simon)
   4. Operational concept. (Simon) – Not needed
4. Implementation.
   1. Software architecture. (Matze) – We can add a copy of the actual class diagram (would be nice)

\_\_\_Stand heute\_\_\_

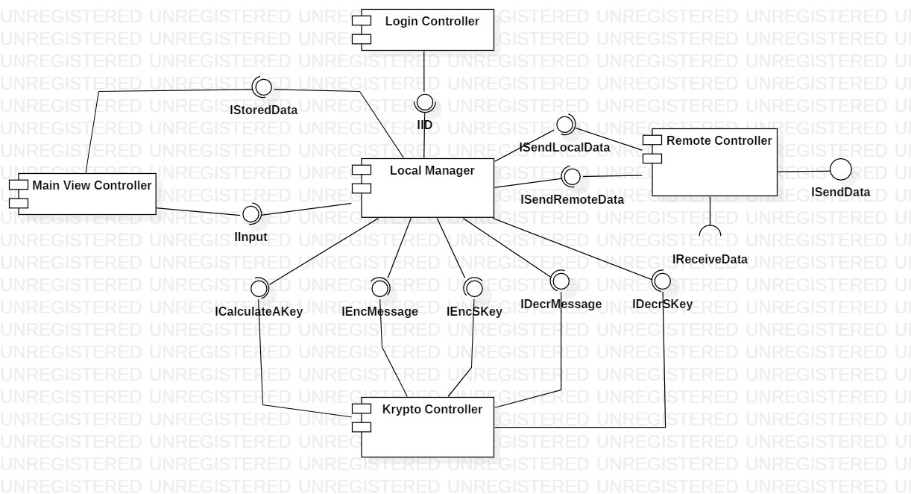
* 1. Software development. (Überschrift)
     1. Theoretical framework (Überschrift) – Not needed
        1. Symmetric Cryptography. (Simon)
        2. Asymmetric Cryptography. (Simon)
        3. Server – Client architecture. (Matze)
        4. . (Observer Pattern usw. Matze )
     2. Software description. (Max)
  2. Instructions of usage. (README.me)
  3. Conclusion. (Max).

***Introduction:***

One of the greatest challenges in undergraduate life is trying to overcome the issues that come together with working as a team. During the university, it is most common for professors to make students create free topic projects in which they have to learn by themselves how to manage their time, organize and distribute work, and more importantly, trust in their teammates; because it is vital for them to learn that it is impossible to do everything by oneself in a working environment. Taking this into account, the Media Systems students of the University of Applied sciences in Hamburg, were tasked with developing a project that made usage of the different skills practiced in previous semesters. This report will shortly explain the making of CryptoChat: an end to end encrypted chat program, with emphasis in the teaching of cryptography, while at the same time showing the practical application of this same premise.

***Concept:***

The first clue for the development of CryptoChat was going to be was born by the students’ curiosity towards cryptography. Not knowing what exactly they wanted to do, but at the same time craving to improve their knowledge in the fields of cryptography, they were set up to create something that was based on these principles. The aforementioned idea was, for the most part, the core of the whole project, and was always to be maintained as such, realizing the project around it. Because of this, the concept ended up being inspired by the modern-day chat software, which already utilize different type of encryption, but having on the counter side the big question of: What if you could decide your own encryption, while at the same time understanding what is happening behind the curtains?

In order to begin with the implementation, one of the first questions in a developer mind has to be the programming language in which a program is going to be written, but that was something that the students settled easily; the answer was Java. The decision for this was the ease of access and experience the developers had with this specific language, making it a clear cut in comparison with other languages like C++. Moreover, it was set that the program was not going to be a web application, which in turn got rid of options like JavaScript or HTML5. It is important to explain, that the creators wanted to use the least amount of external libraries as possible, meaning that everything they programmed was, for the most part, “from scratch”.

Looking at the component diagram to the right, it is possible to understand the initial planning for the program. It was supposed to have a local manager component which was going to be in charge of the supervision of all the data that came into a client, while at the same time distributing tasks for the rest of the local components to process. This local manager controller was also thought to be able to connect to a remote controller component, which would be obviously resting on the server, and was in turn there for the successful communication between clients. This aforementioned controller was then also made to be the universal storage for the public keys of each and every one of the connected clients, in order for other clients to have an ease of access to these when needing to encrypt a message.

1. Initial Component Diagram

***Implementation:***

***Conclusion:***